PCT WORLD INTELLECTUAL PROPERTY ORGANIZATION
International Bureau

INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

(51) International Patent Classification 6 : H03M 7/30, H04L 23/00, G06F 7/00, 7/06, 7/22 — A1

(11) International Publication Number: **WO 96/25801**

(43) International Publication Date: 22 August 1996 (22.08.96)

(21) International Application Number: PCT/AU96/00081

(22) International Filing Date: 15 February 1996 (15.02.96)

(30) Priority Data:

PN 1232	17 February 1995 (17.02.95)	AU
PN 2392	12 April 1995 (12.04.95)	AU

(71) Applicant *(for all designated States except US)*: TRUSTUS PTY. LTD. [AU/AU]; 200 East Terrace, Adelaide, S.A. 5000 (AU).

(72) Inventor; and

(75) Inventor/Applicant *(for US only)*: WILLIAMS, Ross, Neil [AU/AU]; 16 Lerwick Avenue, Hazelwood Park, S.A. 5066 (AU).

(74) Agent: MADDERNS; 1st floor, 64 Hindmarsh Square, Adelaide, S.A. 5000 (AU).

(81) Designated States: AL, AM, AT, AU, AZ, BB, BG, BR, BY, CA, CH, CN, CZ, DE, DK, EE, ES, FI, GB, GE, HU, IS, JP, KE, KG, KP, KR, KZ, LK, LR, LS, LT, LU, LV, MD, MG, MK, MN, MW, MX, NO, NZ, PL, PT, RO, RU, SD, SE, SG, SI, SK, TJ, TM, TR, TT, UA, UG, US, UZ, VN, ARIPO patent (KE, LS, MW, SD, SZ, UG), Eurasian patent (AZ, BY, KG, KZ, RU, TJ, TM), European patent (AT, BE, CH, DE, DK, ES, FR, GB, GR, IE, IT, LU, MC, NL, PT, SE), OAPI patent (BF, BJ, CF, CG, CI, CM, GA, GN, ML, MR, NE, SN, TD, TG).

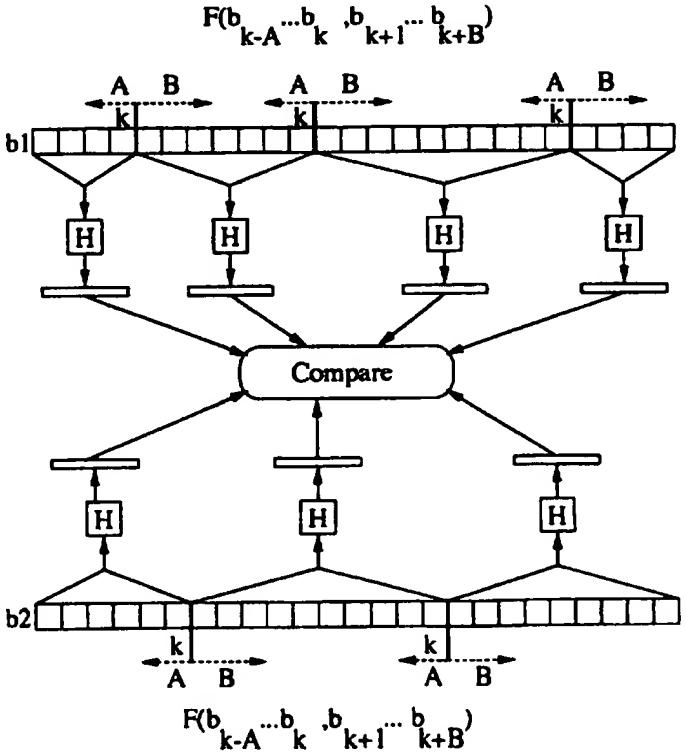
Published

With international search report.
With amended claims and statement.

(54) Title: METHOD FOR PARTITIONING A BLOCK OF DATA INTO SUBBLOCKS AND FOR STORING AND COMMUNICATING SUCH SUBBLOCKS

(57) Abstract

This invention provides a method and apparatus for detecting common spans within one or more data blocks by partitioning the blocks (figure 4) into subblocks and searching the group of subblocks (figure 12) (or their corresponding hashes (figure 13)) for duplicates. Blocks can be partitioned into subblocks using a variety of methods, including methods that place subblock boundaries at fixed positions (figure 3), methods that place subblock boundaries at data-dependent positions (figure 3), and methods that yield multiple overlapping subblocks (figure 6). By comparing the hashes of subblocks, common spans of one or more blocks can be identified without ever having to compare the blocks or subblocks themselves (figure 13). This leads to several applications including an incremental backup system that backs up changes rather than changed files (figure 25), a utility that determines the similarities and differences between two files (figure 13), a file system that stores each unique subblock at most once (figure 26), and a communications system that eliminates the need to transmit subblocks already possessed by the receiver (figure 19).



FOR THE PURPOSES OF INFORMATION ONLY

Codes used to identify States party to the PCT on the front pages of pamphlets publishing international applications under the PCT.

AM	Armenia	GB	United Kingdom	MW	Malawi
AT	Austria	GE	Georgia	MX	Mexico
AU	Australia	GN	Guinea	NE	Niger
BB	Barbados	GR	Greece	NL	Netherlands
BE	Belgium	HU	Hungary	NO	Norway
BF	Burkina Faso	IE	Ireland	NZ	New Zealand
BG	Bulgaria	IT	Italy	PL	Poland
BJ	Benin	JP	Japan	PT	Portugal
BR	Brazil	KE	Kenya	RO	Romania
BY	Belarus	KG	Kyrgystan	RU	Russian Federation
CA	Canada	KP	Democratic People's Republic of Korea	SD	Sudan
CF	Central African Republic			SE	Sweden
CG	Congo	KR	Republic of Korea	SG	Singapore
CH	Switzerland	KZ	Kazakhstan	SI	Slovenia
CI	Côte d'Ivoire	LI	Liechtenstein	SK	Slovakia
CM	Cameroon	LK	Sri Lanka	SN	Senegal
CN	China	LR	Liberia	SZ	Swaziland
CS	Czechoslovakia	LT	Lithuania	TD	Chad
CZ	Czech Republic	LU	Luxembourg	TG	Togo
DE	Germany	LV	Latvia	TJ	Tajikistan
DK	Denmark	MC	Monaco	TT	Trinidad and Tobago
EE	Estonia	MD	Republic of Moldova	UA	Ukraine
ES	Spain	MG	Madagascar	UG	Uganda
FI	Finland	ML	Mali	US	United States of America
FR	France	MN	Mongolia	UZ	Uzbekistan
GA	Gabon	MR	Mauritania	VN	Viet Nam

Method For Partitioning A Block of Data Into Subblocks And For Storing And Communicating Such Subblocks

INTRODUCTION

The present invention provides a method and apparatus for identifying identical subblocks of data within one or more blocks of data and of communicating and storing such subblocks in an efficient manner.

BACKGROUND

Much the massive amount of information stored, communicated, and manipulated by modern computer systems is duplicated within the same or a related computer system. It is commonplace, for example, for computers to store many slightly differing versions of the same document. It is also commonplace for data transmitted during a backup operation to be almost identical to the data transmitted during the previous backup operation. Computer networks also must repeatedly carry the same or similar data in accordance the requirements of their users.

Despite the obvious benefits that would flow from a reduction in the redundancy of communicated and stored data, few computer systems perform any such optimization. Some instances can be found at the application level, one example being the class of incremental backup utilities that save only those files that have changed since the most recent backup. However, even these utilities do not attempt to exploit the significant similarities between old and new versions of files, and between

files sharing other close semantic ties. This kind of redundancy can be approached only by analysing the contents of the files.

The present invention addresses the potential for reducing redundancy by providing an efficient method for identifying identical portions of data within a group of blocks of data, and for using this identification to increase the efficiency of systems that store and communicate data.

SUMMARY OF THE INVENTION

To identify identical portions of data within a group of blocks of data, the blocks must be analysed. In a simple aspect of the invention, the blocks are divided into fixed-length (e.g. 512-byte) subblocks and these subblocks are compared with each other so as to identify all identical subblocks. This knowledge can then be used to manage the blocks in more efficient ways.

Unfortunately, the partitioning of blocks into fixed-length subblocks does not always provide a suitable framework for the recognition of duplicated portions of data, as identical portions of data can occur in different sizes and places within a group of blocks of data. Figure 1 shows how division into fixed-size subblocks fails to generate identical subblocks in two blocks whose only difference is the insertion of a single byte ('X'). A comparison of the two groups of subblocks would reveal no identical pairs of subblocks.

In a more sophisticated aspect of the invention, the blocks are partitioned at boundaries determined by the content of the data itself. For example, the block could be divided at each point at which the preceding three bytes hash to a particular constant value. Figure 2 shows how such a partitioning could turn out, and contrasts it with a fixed-length partitioning.

The fact that a partitioning is data dependent does not imply that it must incorporate any knowledge of the syntax or semantics of the data. So long as the boundaries are positioned in a manner dependent on the local data content, identical subblocks are likely to be formed from identical portions of data, even if the two portions are not identically aligned relative to the start of their enclosing blocks (Figure 3).

Once the group of blocks has been partitioned into subblocks, the resulting group of subblocks can be manipulated in a manner that exploits the occurrence of duplicate subblocks. This leads to a variety of applications, some of which are listed below. However, the application of a further aspect of the invention leads to even greater benefits.

In a further aspect of the invention, the hash of one or more subblocks is calculated. The hash function can be an ordinary hash function or one providing cryptographic strength. The hash function maps each subblock into a small tractable value (e.g. 128 bits) that provides an identity of the subblock. These hashes can usually be manipulated more efficiently than their corresponding subblocks.

Some applications of aspects of this invention are:

Fine-grained incremental backups: Conventional incremental backup technology uses the file as the unit of backup. However, in practice many large files change only slightly, resulting in a wasteful re-transmission of changed files. By storing the hashes of subblocks of the previous versions of files, the transmission of unchanged subblocks can be eliminated.

Communications: By providing a framework for communicating the hashes of subblocks, the invention can eliminate the transmission of subblocks already possessed by the receiver.

Differences: The invention could be used as the basis of a program that determines the areas of similarity and difference between two blocks.

Low-redundancy file system: Data stored in a file system can be partitioned into subblocks whose hashes can be compared so as to eliminate the redundant storage of identical subblocks.

Virtual memory: Virtual memory could be organized by subblock using a table of hashes to determine if a subblock is somewhere in memory.

Clarification Of Terms

The term **block** and **subblock** both refer, without limitation, to finite blocks or infinite blocks (sometimes called streams) of zero or more bits or bytes of digital data. Although the two different terms ("block" and "subblock") essentially describe the same substance (digital data), the two different terms have been employed so as to indicate the role that a particular piece of data is playing. The term "block" is usually used to refer to raw data to be manipulated by aspects of the invention. The term "subblock" is usually used to refer to a part of a block.

The term **partition** has its usual meaning of exhaustively dividing an entity into mutually exclusive parts. However, within this patent specification, the term also includes:

- Analyses in which only one or more parts are analysed.
- Analyses in which multiple overlapping subblocks are formed.

A **natural number** is a non-negative integer (0, 1, 2, 3, 4, 5, ...).

Where the phrase **zero or more** is used. this phrase is intended to encompass the degenerate case where the objects being enumerated are not considered at all, as well as the case where zero or more objects are used.

BRIEF DESCRIPTION

The following aspects of this invention are numbered for reference purposes. The terms "block" and "subblock" refer to blocks and subblocks of digital data.

1. In an aspect of the invention, the invention provides a method for partitioning a block b into one or more subblocks, the method using the component:

(i) a deterministic or non-deterministic function F that returns one of at least two values, and whose arguments include at least a block of A bits and a block of B bits, where A and B are natural numbers;

and comprising the step of:

a. Basing the positions of subblock boundaries on the positions k in the block for which $F(b_{k-A} \ldots b_k, b_{k+1} \ldots b_{k+B})$ falls within a predetermined subclass of the set of possible function result values.

Note: The specification of this aspect (and each other specification of an aspect involving a function F) encompasses the degenerate case in which either A or B is zero and the function takes (without limitation) just one of the two arguments described. Such specifications also include the case of functions F that do not use some bits of their arguments. A function F that bases its calculation solely on (say) b_{k-3} and b_{k+2} would fall under the classes of F constrained by the condition $A \geq 3$ and $B \geq 2$.

2. In a further aspect of the invention, the invention provides a method for locating the nearest subblock boundary on a particular side of a particular position p within

a block. the method using the same components as aspect 1, but replacing step (a) with:

a. Evaluating $F(b_{p-A} \ldots b_p, b_{p+1} \ldots b_{p+B})$ for increasing (or decreasing) p until the result of F falls within a predetermined subclass of the set of possible function result values, the position of the resultant boundary being based on this position.

3. In a further aspect of the invention, the invention provides a method for partitioning a block into one or more subblocks, the method being identical to one of those above. wherein boundaries may be added and removed in accordance with a further method.

4. In a further aspect of the invention, the invention provides a method for partitioning a block into one or more subblocks, the method being identical to one of those above. wherein an upperbound U on the subblock size is imposed.

5. In a further aspect of the invention. the invention provides a method for partitioning a block into one or more subblocks, the method being identical to one of those above. wherein a lowerbound L on the subblock size is imposed.

6. In a further aspect of the invention, the invention provides a method for partitioning a block into one or more subblocks, the method being identical to one of those above. wherein an upperbound U on the subblock size is imposed and a lowerbound L on the subblock size is also imposed.

7. In a further aspect of the invention. the invention uses one or more of the methods above. but applies more than one partitioning function (e.g. F_1. F_2. ...) and method independently to the block b so as to form more than one group of subblocks.

Note: The subblocks of the various groups are very likely to overlap. The groups produced by this aspect can be used independently or combined in various ways to form larger groups of subblocks.

8. In a further aspect of the invention, the invention provides a method for partitioning a block into one or more subblocks by dividing the block into subblocks of equal size.

Note: This aspect is not novel and has been included solely so that later aspects can refer to it.

9. In a further aspect of the invention, the invention provides a method for partitioning a block into one or more subblocks by dividing the block into subblocks of a small number of different sizes.

Note: This aspect is not novel and has been included solely so that later aspects can refer to it.

10. In a further aspect of the invention, the invention uses one of the methods above, and additionally forms subblocks from one or more groups of subblocks.

11. In a further aspect of the invention, the invention employs one of the methods above, and additionally forms a hierarchy of subblocks from one or more contiguous groups of subblocks.

Note: The aspects above will be referred to as the **partitioning aspects.**

12. In a further aspect of the invention, the invention provides a method for partitioning a block into subblocks and forming a corresponding collection of hashes, comprising the steps of:

a. Partitioning the block into one or more subblocks in accordance with any partitioning aspect;

b. Calculating the hash of one or more subblocks using a hash function H.

Note: The collection of hashes is particularly useful if H is a strong one-way hash function.

13. In a further aspect of the invention, the invention provides a method for constructing a projection of a block, comprising the steps of:

a. Partitioning the block into one or more subblocks in accordance with any partitioning aspect;

b. Forming a projection which is an ordered or unordered list containing identities (e.g. subblocks or hashes of subblocks) of, or references to, one or more of the subblocks.

Note: The specification of this aspect is intended to admit lists that contain a mixture of various kinds of identities and references.

Note: In most applications the output of this aspect will be an ordered list of hashes of the subblocks of the block.

14. In a further aspect of the invention, the invention provides a method for finding identical portions within a group of one or more blocks comprising the steps of:

a. Partitioning one or more of said blocks into one or more subblocks in accordance with an aspect above;

b. Comparing the subblocks or the identities (e.g. hashes) of the subblocks.

15. In a further aspect of the invention, the invention provides a method for representing one or more blocks, involving the following components:

(i) A method for storing and retrieving subblocks;

(ii) A mapping from block representatives (e.g. filenames) to lists of entries that identify subblocks;

whereby the modification of data in a stored block involves the following steps:

a. Partitioning the new data into subblocks in accordance with any partitioning aspect:

b. Adding subblocks in the new data that are not already in the collection of stored subblocks to the collection of stored subblocks, and updating the subblock list associated with the block being modified;

16. In a further aspect of the invention, the invention provides a method for an entity $E1$ to communicate a group X of one or more subblocks $X_1 \ldots X_n$ to $E2$ where $E1$ possesses the knowledge that $E2$ possesses a group Y of zero or more subblocks $Y_1 \ldots Y_m$ comprising the following step:

a. Transmitting from $E1$ to $E2$ the contents of a subset of zero or more subblocks in X, and the remaining subblocks as references which may take (but are not limited to) the following forms:

(i) a hash of a subblock;

(ii) a reference to a subblock in Y;

(iii) a reference to a range of subblocks in Y;

(iv) a reference to a subblock already transmitted;

(v) a reference to a range of subblocks already transmitted.

Note: In most implementations of this aspect, the subblocks whose contents are transmitted will be those in X that are not in Y, and for which no identical subblock has previously been transmitted.

Note: To possess knowledge that $E2$ possesses $Y_1 \ldots Y_m$, $E1$ need not actually possess $Y_1 \ldots Y_m$ itself. $E1$ need only possess the identities of $Y_1 \ldots Y_m$ (e.g. the hashes of each

subblock $Y_1 \ldots Y_m$). This specification is intended to admit any other representation in which $E1$ may have the knowledge that $E2$ possesses (or has access to) $Y_1 \ldots Y_m$. In particular, the knowledge may take the form of a projection of Y.

Note: It is implicit in this aspect that $E1$ will be able to use comparison (or other methods) to use its knowledge of $E2$'s possession of Y to determine the set of subblocks that are common to both X and Y. For example, if $E1$ possessed the hashes of the subblocks of Y, it could compare them to the hashes of the subblocks of X to determine the subblocks common to both X and Y. Subblocks that are not common can be transmitted explicitly. Subblocks that are common to both X and Y can be transmitted by transmitting a reference to the subblock.

17. In a further aspect of the invention, the invention provides a method for an entity $E1$ to communicate a block X to $E2$ where $E1$ possesses the knowledge that $E2$ possesses a group Y of subblocks $Y_1 \ldots Y_m$ comprising step (a) of aspect 16 preceded by the step:

s. Partitioning X into subblocks $X_1 \ldots X_n$ in accordance with any partitioning aspect.

18. In a further aspect of the invention, the invention provides a method for an entity $E1$ to communicate one or more subblocks of a group X of subblocks $X_1 \ldots X_n$ to $E2$ where $E1$ possesses the knowledge that $E2$ possesses the block Y, comprising step (a) of aspect 16 preceded by the step:

s. Partitioning Y into subblocks $Y_1 \ldots Y_m$ in accordance with any partitioning aspect.

19. In a further aspect of the invention, the invention provides a method for an entity $E1$ to communicate a block X to $E2$ where $E1$ possesses the knowledge that $E2$ possesses block Y, comprising step (a) of aspect 16 preceded by the steps:

s1. Partitioning X into subblocks $X_1 \ldots X_n$ in accordance with any partitioning aspect.

s2. Partitioning Y into subblocks $Y_1 \ldots Y_m$ in accordance with any partitioning aspect.

Note: Steps (s1) and (s2) could be performed in any order, or concurrently, as could many other subsets of steps in this patent specification.

20. In a further aspect of the invention, the invention provides a method for constructing a block D from a group X of one or more subblocks $X_1 \ldots X_n$ and a group Y of zero or more subblocks $Y_1 \ldots Y_m$ such that X can be constructed from Y and D, comprising the step:

a. Constructing D from at least one of the following components:

(1) the contents of one or more subblocks in X;

(2) references to subblocks in Y or to subblocks included in D, or to a range of subblocks from either D or Y.

Note: Component 2 above is intended to encompass the case where a mixture of the elements it describes is used.

21. In a further aspect of the invention, the invention provides a method for constructing a block D from a block X and a group Y of subblocks $Y_1 \ldots Y_m$ such that X can be constructed from Y and D, comprising step (a) of aspect 20 preceded by the step:

s. Partitioning X into subblocks $X_1 \ldots X_n$ in accordance with any partitioning aspect.

22. In a further aspect of the invention, the invention provides a method for constructing a block D from a group X of subblocks $X_1 \ldots X_n$ and a block Y such that X can be constructed from Y and D, comprising step (a) of aspect 20 preceded by the step:

s. Partitioning Y into subblocks $Y_1 \ldots Y_m$ in accordance with any partitioning aspect.

23. In a further aspect of the invention, the invention provides a method for constructing a block D from a block X and a block Y such that X can be constructed from Y and D, comprising step (a) of aspect 20 preceded by the steps:

s1. Partitioning X into subblocks $X_1 \ldots X_n$ in accordance with any partitioning aspect.

s2. Partitioning Y into subblocks $Y_1 \ldots Y_m$ in accordance with any partitioning aspect.

Note: Steps (s1) and (s2) could be performed in any order, or concurrently, as could many other subsets of steps in this patent specification.

24. In a further aspect of the invention, the invention provides a method for constructing a block D from a group X of subblocks $X_1 \ldots X_n$ and a projection of a block Y (or a projection of a group Y of subblocks $Y_1 \ldots Y_m$), such that X can be constructed from Y and D, comprising the step:

a. Constructing D from at least one of the following components:

(1) the contents of one or more subblocks in X;

(2) references to subblocks in Y or to subblocks included in D, or to a range of subblocks from either D or Y.

Note: The projection of Y will usually have been calculated in accordance with aspect 13. The projection of a group of subblocks will usually have been calculated in accordance with step (b) of aspect 13.

Note: An implementation will usually be able to use the projection of Y to determine if a subblock in X is also in Y.

Note: Component 2 above is intended to encompass the case where a mixture of the elements it describes is used.

25. In a further aspect of the invention, the invention provides a method for constructing a block D from a block X and a projection of Y such that X can be constructed from Y and D, comprising the step of aspect 24 with the following step inserted before step (a):

s. Partitioning X into subblocks $X_1 \ldots X_n$ in accordance with any partitioning aspect;

26. In a further aspect of the invention, the invention provides a method for constructing a block X (or group X of subblocks $X_1 \ldots X_n$) from a group Y of subblocks $Y_1 \ldots Y_m$ and a block D, where D was constructed in accordance with one of aspects 20 to 25 above, comprising the step of:

a. Constructing X from D and Y by constructing the subblocks of X based on one or more of:

(i) references in D to subblocks in Y:

(ii) references in D to subblocks in D:

(iii) references in D that specify a range of subblocks in Y:

(iv) references in D that specify a range of subblocks in D:

(v) subblocks contained within D;

(vi) other data elements in D.

27. In a further aspect of the invention, the invention provides a method for constructing a block X (or group X of subblocks $X_1 \ldots X_n$) from a block Y and a block D, where D was constructed in accordance with one of aspects 20 to 25 above, comprising the step of aspect 26 with the following step inserted before step (a):

s. Partitioning Y into subblocks $Y_1 \ldots Y_m$ in accordance with any partitioning aspect.

28. In a further aspect of the invention, the invention provides a method for transmitting a group X of subblocks $X_1 \ldots X_n$ from one entity $E1$ to another entity $E2$, comprising the steps of:

a. Transmitting from $E1$ to $E2$ an identity of one or more subblocks;

b. Transmitting from $E2$ to $E1$ information communicating the presence or absence of subblocks at $E2$;

c. $E1$ transmitting to $E2$ at least the subblocks identified in step (b) as not being present at $E2$:

Note: The information communicated in step (b) could take the form of a bitmap (or a compressed bitmap) corresponding to the subblocks referred to in step (a). It could also take many other forms.

Note: If a group of subblocks are to be transmitted, the above steps could be performed completely for each subblock before moving onto the next subblock. The steps could be applied to any subgroup of subblocks.

29. In a further aspect of the invention, the invention provides a method for communicating a data block X from one entity $E1$ to another entity $E2$, comprising the steps of aspect 28 but with the following step inserted before step (a):

s. Partitioning X into subblocks $X_1 \ldots X_n$ in accordance with any partitioning aspect.

30. In a further aspect of the invention, the invention provides a method for comparing the contents of two or more blocks comprising the steps:

a. Constructing a projection of each block as described in aspect 13;

b. Comparing the projections of the blocks.

Note: The phrase "comparing the projections" is intended to include not just the case where the two projections are tested to see if they are the same, but also the case where the subblocks (or projections of subblocks) within the projections are compared so as to determine the subblocks that are common to the two original blocks.

31. In a further aspect of the invention, the invention provides a method for transmitting a group X of subblocks $X_1 \ldots X_n$ from one entity $E1$ to another entity $E2$, comprising the steps of:

a. Transmitting from $E2$ to $E1$ information communicating the presence or absence at $E2$ of members of a group Y of subblocks $Y_1 \ldots Y_m$;

b. Transmitting from $E1$ to $E2$ the contents of zero or more subblocks in X, and the remaining subblocks as references which may take (but are not limited to) the following forms:

(i) a hash of a subblock;

(ii) a reference to a subblock in Y;

(iii) a reference to a range of subblocks in Y;

(iv) a reference to a subblock already transmitted;

(v) a reference to a range of subblocks already transmitted.

Note: The information communicated in step (a) could take the form of subblock identities such as hashes. It could also take many other forms.

32. In a further aspect of the invention, the invention provides a method for transmitting a block X from one entity $E1$ to another entity $E2$, comprising the steps of aspect 31 with the following step inserted before step (a):

s. $E1$ partitioning X into subblocks $X_1 \ldots X_n$ in accordance with any partitioning aspect;

33. In a further aspect of the invention, the invention provides a method for an entity $E2$ to communicate to an entity $E1$ the fact that $E2$ possesses a group Y of subblocks $Y_1 \ldots Y_m$, comprising the step of:

a. $E2$ transmitting to $E1$ identities or references of the subblocks $Y_1 \ldots Y_m$.

34. In a further aspect of the invention, the invention provides a method for an entity $E2$ to communicate to an entity $E1$ the fact that $E2$ possesses a block Y, comprising the step of aspect 33 with the following step inserted before step (a):

s. $E2$ partitioning Y into subblocks $Y_1 \ldots Y_m$ in accordance with any partitioning aspect;

35. In a further aspect of the invention, the invention provides a method for an entity $E1$ to communicate a subblock X_i to an entity $E2$, comprising the steps:

a. $E2$ sending $E1$ an identity of X_i.

b. $E1$ sending X_i to $E2$.

Note: This aspect applies (among other applications) to the case of a network server $E1$ that serves subblocks to clients such as $E2$, given the identities (e.g. hashes) of the requested subblocks.

36. In a further aspect of the invention, the invention provides a method for an entity $E1$ to communicate a subblock X_i to an entity $E2$, comprising the steps of aspect 35 with the following step inserted before step (a):

s. $E1$ partitioning a block X into subblocks $X_1 \ldots X_n$ in accordance with any partitioning aspect;

37. In a further aspect of the invention, the invention provides a method similar to any of those above, but where one or more of the comparisons of subblocks are performed by comparing the hashes of the subblocks, using hashes already available (e.g. as a byproduct of other steps), or calculated for the purpose of performing one or more said comparisons.

38. In a further aspect of the invention the invention provides a method similar to any of those above, but where subsets of identical subblocks within a group of one or more subblocks are identified by inserting each subblock, an identity of each subblock, a reference of each subblock, or a hash of each subblock, into a data structure.

39. In a further aspect of the invention the invention provides a method identical to any of those above, but with various subsets of steps executed concurrently.

40. In an aspect of the invention, the invention provides an apparatus for partitioning a block b into one or more subblocks, the apparatus comprising:

(i) means for evaluating a deterministic or non-deterministic function F that returns

one of at least two values, and whose arguments include at least a block of A bits and a block of B bits. where A and B are natural numbers;

and comprising the step of

a. Generating a set of partitions of b, basing these upon the positions of subblock boundaries on the positions k in the block for which $F(b_{k-A} \ldots b_k, b_{k+1} \ldots b_{k+B})$ falls within a predetermined subclass of the set of possible function result values.

Note: The specification of this aspect (and each other specification of an aspect involving a function F) encompasses the degenerate case in which either A or B is zero and the function takes (without limitation) just one of the two arguments described. Such specifications also include the case of functions F that do not use some bits of their arguments. A function F that bases its calculation solely on (say) b_{k-3} and b_{k+2} would fall under the classes of F constrained by the condition $A \geq 3$ and $B \geq 2$.

41. In a further aspect of the invention, the invention provides an apparatus for locating the nearest subblock boundary on a particular side of a particular position p within a block b. the apparatus comprising the same elements as aspect 40, but replacing step (a) with

a. Generating a position within b by evaluating $F(b_{p-A} \ldots b_p. b_{p+1} \ldots b_{p+B})$ for increasing (or decreasing) p until the result of F falls within a predetermined subclass of the set of possible function result values. the position being based on this position.

42. In a further aspect of the invention. the invention provides an apparatus for partitioning a block into one or more subblocks comprising

(i) means for dividing a block into subblocks of equal size.

Note: This aspect is not novel and has been included solely so that later aspects can refer to it.

43. In a further aspect of the invention, the invention provides an apparatus for partitioning a block into one or more subblocks comprising

(i) means for dividing a block into subblocks of a small number of different sizes.

Note: This aspect is not novel and has been included solely so that later aspects can refer to it.

44. In a further aspect of the invention, the invention provides an apparatus $E1$ that can communicate a group X of one or more subblocks $X_1 \ldots X_n$ to an entity $E2$ where $E1$ possesses the knowledge that $E2$ possesses a group Y of zero or more subblocks $Y_1 \ldots Y_m$, the apparatus comprising

(i) means for manipulating subblocks, subblock identities, and subblock references:

and comprising the step

a. Transmitting from $E1$ to $E2$ the contents of a subset of zero or more subblocks in X, and the remaining subblocks as references which may take (but are not limited to) the following forms:

(i) a hash of a subblock;

(ii) a reference to a subblock in Y;

(iii) a reference to a range of subblocks in Y;

(iv) a reference to a subblock already transmitted:

(v) a reference to a range of subblocks already transmitted.

Note: In most implementations of this aspect, the subblocks whose contents are transmitted will be those in X that are not in Y, and for which no identical subblock has previously been transmitted.

Note: To possess knowledge that E2 possesses $Y_1 \ldots Y_m$, E1 need not actually possess $Y_1 \ldots Y_m$ itself. E1 need only possess the identities of $Y_1 \ldots Y_m$ (e.g. the hashes of each subblock $Y_1 \ldots Y_m$). This specification is intended to admit any other representation in which E1 may have the knowledge that E2 possesses (or has access to) $Y_1 \ldots Y_m$. In particular, the knowledge may take the form of a projection of Y.

Note: It is implicit in this aspect that E1 will be able to use comparison (or other methods) to use its knowledge of E2's possession of Y to determine the set of subblocks that are common to both X and Y. For example, if E1 possessed the hashes of the subblocks of Y, it could compare them to the hashes of the subblocks of X to determine the subblocks common to both X and Y. Subblocks that are not common can be transmitted explicitly. Subblocks that are common to both X and Y can be transmitted by transmitting a reference to the subblock.

45. In a further aspect of the invention, the invention provides an apparatus for constructing a block D from a group X of one or more subblocks $X_1 \ldots X_n$ and a group Y of zero or more subblocks $Y_1 \ldots Y_m$ such that X can be constructed from Y and D, the apparatus comprising:

(i) means for manipulating subblocks, subblock identities and subblock references;

and comprising the step

a. Constructing D from at least one of the following components:

(1) the contents of one or more subblocks in X;

(2) references to subblocks in Y or to subblocks included in D, or to a range of subblocks from either D or Y.

Note: Component 2 above is intended to encompass the case where a mixture of the elements it describes is used.

46. In a further aspect of the invention, the invention provides an apparatus for constructing a block D from a group X of subblocks $X_1 \ldots X_n$ and a projection of a block Y (or a projection of a group Y of subblocks $Y_1 \ldots Y_m$), such that X can be constructed from Y and D, the apparatus comprising

(i) means for manipulating subblocks, subblock identities and subblock references;

and comprising the step:

a. Constructing D from at least one of the following components:

(1) the contents of one or more subblocks in X:

(2) references to subblocks in Y or to subblocks included in D, or to a range of subblocks from either D or Y.

Note: The projection of Y will usually have been calculated in accordance with aspect 13. The projection of a group of subblocks will usually have been calculated in accordance with step (b) of aspect 13.

Note: An implementation will usually be able to use the projection of Y to determine if a subblock in X is also in Y.

Note: Component 2 above is intended to encompass the case where a mixture of the elements it describes is used.

47. In a further aspect of the invention, the invention provides an apparatus for constructing a block X (or group X of subblocks $X_1 \ldots X_n$) from a group Y of subblocks $Y_1 \ldots Y_m$ and a block D, where D was constructed in accordance with one of aspects 20 to 25 above, the apparatus comprising

(i) means for manipulating subblocks, subblock identities and subblock references:

and comprising the step of

a. Constructing X from D and Y by constructing the subblocks of X based on one or more of:

(i) references in D to subblocks in Y;

(ii) references in D to subblocks in D;

(iii) references in D that specify a range of subblocks in Y:

(iv) references in D that specify a range of subblocks in D:

(v) subblocks contained within D:

(vi) other data elements in D.

48. In a further aspect of the invention, the invention provides a system for transmitting a group X of subblocks $X_1 \ldots X_n$ from one apparatus $E1$ to another apparatus $E2$. each apparatus consisting of

(i) means for manipulating subblocks, subblock identities and subblock references:

and the systems execution comprising the steps of:

a. Transmitting from $E1$ to $E2$ an identity of one or more subblocks:

b. Transmitting from $E2$ to $E1$ information communicating the presence or absence of subblocks at $E2$:

c. $E1$ transmitting to $E2$ at least the subblocks identified in step (b) as not being present at $E2$:

Note: The information communicated in step (b) could take the form of a bitmap (or a compressed bitmap) corresponding to the subblocks referred to in step (a). It could also take many other forms.

Note: If a group of subblocks are to be transmitted, the above steps could be performed completely for each subblock before moving onto the next subblock. The steps could be applied to any subgroup of subblocks.

49. In a further aspect of the invention, the invention provides a system for transmitting a group X of subblocks $X_1 \ldots X_n$ from one apparatus $E1$ to another apparatus $E2$, each apparatus consisting of

(i) means for manipulating subblocks, subblock identities and subblock references:

and comprising the steps of:

a. Transmitting from $E2$ to $E1$ information communicating the presence or absence at $E2$ of members of a group Y of subblocks $Y_1 \ldots Y_m$;

b. Transmitting from $E1$ to $E2$ the contents of zero or more subblocks in X, and the remaining subblocks as references which may take (but are not limited to) the following forms:

(i) a hash of a subblock;

(ii) a reference to a subblock in Y;

(iii) a reference to a range of subblocks in Y;

(iv) a reference to a subblock already transmitted;

(v) a reference to a range of subblocks already transmitted.

Note: The information communicated in step (a) could take the form of subblock identities such as hashes. It could also take many other forms.

50. In a further aspect of the invention, the invention provides a system for an apparatus $E2$ to communicate to an apparatus $E1$ the fact that $E2$ possesses a group Y of subblocks $Y_1 \ldots Y_m$, each apparatus consisting of

(i) means for manipulating subblocks, subblock identities and subblock references;

and comprising the step of:

a. $E2$ transmitting to $E1$ identities or references of the subblocks $Y_1 \ldots Y_m$.

51. In a further aspect of the invention, the invention provides a system for an apparatus $E1$ to communicate a subblock X_i to an apparatus $E2$, each apparatus consisting of

(i) means for manipulating subblocks, subblock identities and subblock references;

and comprising the steps:

a. $E2$ sending $E1$ an identity of X_i.

b. $E1$ sending X_i to $E2$.

Note: This aspect applies (among other applications) to the case of a network server $E1$ that serves subblocks to clients such as $E2$, given the identities (e.g. hashes) of the requested subblocks.

BRIEF DESCRIPTION OF FIGURES

Figure 1 shows how data can become "misaligned" relative to its containing blocks when data in inserted.

Figure 2 shows how data can be divided into fixed-width subblocks or variable-width subblocks.

Figure 3 shows how data-dependent partitions move with the data when the data is shifted (e.g. by an insertion) (Compare with Figure 1).

Figure 4 depicts the data-dependent partitioning of a block of data b into subblocks using a function F.

Figure 5 depicts the search within a block b for a subblock boundary (as defined by F) using F.

Figure 6 shows how a block may be subdivided in different ways using different boundary functions.

Figure 7 shows how "higher order" subblocks can be constructed from one or more initial subblocks.

Figure 8 shows how different partitioning functions can produce subblocks of differing average sizes.

Figure 9 shows how subblocks can be constructed (or organized) into a hierarchy. Such a hierarchy can be constructed by restricting in stages, F's "partition" result subset.

Figure 10 depicts a method (and apparatus) for the partitioning of a block b into subblocks using F and the calculation of the hashes of the subblocks using hash function H.

Figure 11 depicts the partitioning of a block b into subblocks using F and the projection of those subblocks into a structure consisting of subblock hashes, subblock data, and subblock references.

Figure 12 depicts a method (and apparatus) for partitioning two blocks $b1$ and $b2$ into subblocks using F and the comparison of those subblocks.

Figure 13 depicts a method (and apparatus) for the partitioning using F of two blocks $b1$ and $b2$ into subblocks, the calculation using H of the hashes of the subblocks, and the comparison of those hashes with each other to determine (among other things) subblocks common to both $b1$ and $b2$.

Figure 14 depicts a method (and apparatus) for a file system that employs an aspect of the invention to eliminate the multiple storage of data common to more than file (or to different parts of the same file).

Figure 15 depicts a method (and apparatus) for the communication of a block X from $E1$ to $E2$ where both $E1$ and $E2$ possess Y.

Figure 16 depicts a method (and apparatus) for the construction of a block D from which X may be later reconstructed, given Y.

Figure 17 depicts a method (and apparatus) for the construction of a block D from which X may be later reconstructed, given Y. In this case, the entity constructing D does not have access to Y, only to a projection of Y (in this case being the hashes of the subblocks of Y).

Figure 18 depicts a method (and apparatus) for the reconstruction of X from the blocks Y and D.

Figure 19 depicts a method (and apparatus ($E1$ and $E2$ at each time)) for the communication of a block X from entity $E1$ to entity $E2$ where $E2$ already possesses Y.

Figure 20 depicts a method (and apparatus ($E1$ and $E2$ at each time)) for the communication of a block X from entity $E1$ to entity $E2$ where $E2$ already possesses Y and where $E2$ first discloses to $E1$ information about Y.

Figure 21 depicts a method (and apparatus) for the communication from entity $E2$ to entity $E1$ information about a block (or group of subblocks) Y' at $E2$.

Figure 22 depicts a method (and apparatus ($E1$ and $E2$ at each time)) for the communication from entity $E1$ to entity $E2$ of subblock X_i following a request by entity $E2$ for the subblock X_i.

Figure 23 depicts an apparatus for partitioning a block b (the input) using a partitioning function F. The output is a set of subblock boundary positions.

Figure 24 depicts a method (and apparatus) for the partitioning of a block b into subblocks using F and the projection of those subblocks into a table of subblock hashes.

Figure 25 depicts a method (and apparatus) for the transmission from entity $E1$ to $E2$ of a block X where $E2$ possesses Y' and $E1$ possesses a table of the hashes of the subblocks of Y' (a projection of Y).

Figure 26 depicts a method (and apparatus) for a file system that employs an aspect of the invention to eliminate the multiple storage of data common to more than file (or to different parts of the same file).

DETAILED DESCRIPTION OF PREFERRED EMBODIMENTS

This section contains a detailed discussion of mechanisms that could be used to implement aspects of the invention. It also contains a selection of examples of implementations of various aspects of the invention. However, nothing in this section should be interpreted as a limitation on the scope of this patent.

Units Of Information

Aspects of this invention can be applied at various levels of granularity of data. For example, if the data was treated as a stream of bits, boundaries could be placed between any two bits. However, if the data was treated as a stream of bytes, boundaries would usually be positioned only between bytes. The invention could be applied with any unit of data, and in this document references to bits and bytes should usually be interpreted as admitting any granularity.

The Concept Of Entity

At various places, this patent specification uses the term "entity" to describe an agent. This term is purposefully vague and is intended to cover all forms of agent including, but not limited to:

- Computer systems.
- Networks of computer systems.
- Processes in computer systems.
- File systems.
- Components of software.
- Dedicated computer systems.
- Communications systems.

The Concepts Of Identity And Reference

This patent specification frequently refers to "identities" of subblocks and "references" to subblocks. These terms are not intended to be defined precisely.

The **identity** of a subblock means any piece of information that could be used in place of the subblock for the purpose of comparison for identicality. Identities include, but are not limited to:

- The subblock itself.
- A hash of the subblock.

The subblock acts as its own identity because subblocks themselves can be compared with each other. Hashes of subblocks also act as identities of subblocks because hashes of subblocks can be compared with each other to determine if their corresponding subblocks are identical.

A **reference** to a subblock means any piece of information that could be used in practice by one entity to identify to another entity (or itself) a particularly valued subblock, where the two entities may already share some kind of knowledge. For example, the two entities might each possess the knowledge that the other entity already possesses ten subblocks of known values having particular index values numbered one to ten.

Once two entities have a basis of shared knowledge, it is possible for them to identify a subblock in ways more concise than the transmission of an identity. A reference to a particularly valued subblock can take (without limitation) the following forms:

- An identity.
- An identifying number of a subblock possessed by the receiver.
- An identifying number of a subblock previously transmitted between the two communicants.

- The location of the block in some shared data space.
- A relative subblock number.
- Ranges of the above.

The concept of **knowledge** of a subblock is related to the concepts of identity and reference. An entity may have knowledge of a subblock (or knowledge that another entity possesses a subblock) without actually possessing the subblock itself. For example, it might possess an identity of the subblock or a reference to the subblock.

The Use Of Ranges

In any situation where a group of values that have contiguous values (e.g. 6, 7, 8, 9) is to be communicated or stored, such a group can be represented using a range (e.g. 6–9) which may take up less communication time or storage space. Ranges can be applied to all kinds of things, such as index values and subblock numbers. In particular, if an entity notices that the references (to subblocks) that it is about to transmit are contiguous, it can replace the references with a range.

Ranges can be represented in any way that identifies the first and last element of the range. Three common representations are:

1. The first and last element of the range.
2. The first element and the length of the range.
3. The last element and the length of the range.

The concept of range can be generalized to include the compression of any group of values that exhibit compressible structure.

The Use Of Backward References

References can be used not only to refer to data shared by two communicants at the start of a transmission, but can also be used to refer to data communicated at some previous time during the transmission.

For example, if an entity A notices that the subblock it is about to transmit to another entity B was not possessed by B at the start of the transmission, but has since been transmitted by A to B, then A could code the second instance of the subblock as a reference to the previous instance of the subblock. The range mechanism can be used here too.

No Requirement For Subblock Framing Information

It should be noted that it is possible that an entity $E1$ could transmit a group X of subblocks $X_1 \ldots X_n$ as a group to an entity $E2$ simply by sending the concatenation of the subblocks. There may be no need for any framing information (e.g. information at the start of each subblock giving the length of the subblock or "escape" codes to indicate subblock boundaries) as $E2$ is capable of partitioning X into $X_1 \ldots X_n$ itself.

No Requirement For Ordering Subblocks

It should be noted that if two entities $E1$ and $E2$ both possess the same *unordered* group Y of subblocks (or knowledge of such a group of subblocks) then even though $E1$ and $E2$ may not possess the subblocks in the same order, the subblocks can still be referred to using a subblock index or serial number. This is achieved by having $E1$ and $E2$ each sort their subblocks in accordance with some mutually agreed (or universally defined) ordering method and then number the subblocks in the resultant ordered group of subblocks. These numbers (or ranges of such numbers) can then be used to refer to subblocks.

An Overview Of Hash Functions

Although the use of a hash function is not essential in all aspects of this invention, hash functions provide such advantages in the implementation of this invention that an overview of them is warranted.

A hash function accepts a variable-length input block of bits and generates an output block of bits that is based on the input block. Most hash functions guarantee that the output block will be of a particular length (e.g. 16 bits) and aspire to provide a random, but deterministic, mapping between the infinite set of input blocks and the finite set of output blocks. The property of randomness enables these outputs, called "hashes", to act as easily manipulated representatives of the original block.

Hash functions come in at least four classes of strength.

Narrow hash functions: Narrow hash functions are the weakest class of hash functions and generate output values that are so narrow (e.g. 16 bits) that the entire space of output values could be searched in a reasonable time. For example, an 8-bit hash function would map any data block to a hash in the range 0 to 255. A 16-bit hash function would map to a hash in the range 0 to 65535. Given a particular hash value, it would be possible to find a corresponding block simply by generating random blocks and feeding them into the narrow hash function until the searched-for value appeared. Narrow hash functions are usually used to arbitrarily (but deterministically) classify a set of data values into a small number of groups. As such, they are useful for constructing hash table data structures, and for detecting errors in data transmitted over noisy communication channels. Examples of this class: CRC-16, CRC-32, Fletcher checksum, the IP checksum.

Wide hash functions: Wide hash functions are similar to narrow hash

functions except that their output values are significantly wider. At a certain point this quantitative difference implies a qualitative difference. In a wide hash function, the output value is so wide (e.g. 128 bits) that the probability of any two randomly chosen blocks having the same hashed value is negligible (e.g. about one in 10^{38}). This property enables these wide hashes to be used as "identities" of the blocks of data from which they are calculated. For example, if entity E1 has a block of data and sends the wide hash of the block to an entity E2, then if entity E2 has a block that has the same hash, then the a-priori probability of the blocks actually being different is negligible. The only catch is that wide hash functions are not designed to be non-invertible. Thus, while the space of (say) 2^{128} values is too large to search in the manner described for narrow hash functions, it may be easy to analyse the hash function and calculate a block corresponding to a particular hash. Accordingly, E1 could fool E2 into thinking E1 had one block when it really had a different block. Examples of this class: any 128-bit CRC algorithm.

Weak one-way hash functions: Weak one-way hash functions are not only wide enough to provide "identity", but they also provide cryptographic assurance that it will be extremely difficult, given a particular hash value, to find a block corresponding to that hash value. Examples of this class: a 64-bit DES hash.

Strong one-way hash functions: Strong one-way hash functions are the same as weak one-way hash functions except that they have the additional property of providing cryptographic assurance that it is difficult to find *any* two different blocks that have the same hash value, where the hash value is unspecified. Examples of this class: MD4, MD5, SHA-1, and Snefru.

These four classes of hash provide a range of hashing strengths from which to choose. As might be expected, the speed of a hash function decreases with strength, providing a tradeoff, and different strengths are appropriate in different applications. However, the difference is small enough to admit the use of strong one-way hash functions in all but the most time-critical applications.

The term **cryptographic hash** is often used to refer to hashes that provide cryptographic strength, encompassing both the class of weak one-way hash functions and the class of strong one-way hash functions. However, as strong one-way hash functions are almost always preferable to weak one-way hash functions, the term "cryptographic hash" is used mainly to refer to the class of strong one-way hash functions.

The present invention can employ hash functions in at least two roles:

1. To determine subblock boundaries.
2. To generate subblock identities.

Depending on the application, hash functions from any of the four classes above could be employed in either role. However, as the determination of subblock boundaries does not require identity or cryptographic strength, it would be inefficient to use hash functions from any but the weakest class. Similarly, the need for identity, the ever-present threat of subversion, and the minor performance penalty for strong one-way hash functions suggests that nothing less than strong one-way hash functions should be used to calculate subblock identities.

The security dangers inherent in employing anything less than a strong one-way hash function to generate identities can be illustrated by considering a communications system or file system that incorporates the invention using any such weaker hash

function. In such a system, an intruder could modify a subblock (to be manipulated by a target system) in such a way that the modified subblock has the same hash as another subblock known by the intruder to be already present in the target system. This could result in the target system retaining its existing subblock rather than replacing it by a new one. Such a weakness could be used (for example) to prevent a target system from properly applying a security patch retrieved over a network.

Thus, while wide hash functions could be safely used to calculate subblocks in systems not exposed to hostile humans, even weak one-way hash functions are likely to be insecure in those systems that are.

We now to turn to the ways in which hashes of blocks or subblocks can actually be used.

The Use Of Cryptographic Hashes

The theoretical properties of cryptographic hashes (and here is meant strong one-way hash functions) yield particularly interesting practical properties. Because such hashes are significantly wide, the probability of two randomly-chosen subblocks having the same hash is practically zero (for a 128-bit hash, it is about one in 10^{38}), and because it is computationally infeasible to find two subblocks having the same hash, it is practically guaranteed that no intelligent agent will be able to do so. The implication of these properties is that from a practical perspective, the finite set of hash values for a particular cryptographic hash algorithm is one-to-one with the infinite set of finite variable length subblocks. This theoretically impossible property manifests itself in practice because of the practical infeasibility of finding two subblocks that hash to the same value.

This property means that, for the purposes of comparison (for identicality), cryptographic hashes may safely be used in place of the subblocks from which they were

calculated. As most cryptographic hashes are only about 128 bits long, hashes provide an extremely efficient way to compare subblocks without requiring the direct comparison of the content of the subblocks themselves. This can be used to eliminate many transmissions of information. For example, a subblock X_1 on a computer $C1$ in Sydney could be compared with a subblock Y_1 on a computer $C2$ in Boston by a computer $C3$ in Paris, with the total theoretical network traffic being just 256 bits ($C1$ and $C2$ each send the 128-bit hash of their respective subblocks to $C3$ for comparison, and $C3$ compares the two hashes).

Some of the ways in which cryptographic hashes could be used in aspects of this invention are:

- Cryptographic hashes can be used to compare two subblocks without having to compare, or requiring access to, the content of the subblocks.
- If it is necessary to be able to determine whether a subblock T is identical to one of a group of subblocks, the subblocks themselves need not be stored, just a list of their hashes. The hash of any candidate subblock can then be compared with the hashes in the list to establish whether the subblock is in the group of subblocks from which the list of hashes was generated.
- Cryptographic hashes can be used to ensure that the partitioning of a block into subblocks and the subsequent reassembly of the subblocks into a reconstructed block is error-free. This can be done by comparing the hash of the original block with the hash of the reconstructed block.
- If an entity $E1$ calculates the hash of a subblock X_1 and transmits it to $E2$, then if $E2$ possesses X_1, or even just the hash of X_1, then $E2$ can determine without any practical doubt that $E1$ possesses $X1$.
- If an entity $E1$ passes a key (consisting of a block of bits) chosen at random to an entity $E2$, $E2$ may then prove to $E1$ that it possesses a subblock by

sending $E1$ the hash of the concatenation of the key and the subblock. This mechanism could be used as an additional check in security applications.

- If a group of subblocks must be compared so as to find all subsets of identical subblocks, the corresponding set of hashes of the subblocks may be calculated and compared instead.

- Many of the uses of cryptographic hashes for subblocks can also be applied to blocks. For example, cryptographic hashes can be used to determine whether a block has changed at all since it was last backed up. Such a check could eliminate the need for further analysis.

Use Of Hashes As A Safety Net

A potential disadvantage of deploying aspects of this invention is that it will add extra complexity to the systems into which it is incorporated. This increased complexity carries the potential to increase the chance of undetected failures.

The main mechanism of complexity introduced by many aspects of the invention is the partitioning of blocks (e.g. files) into subblocks, and the subsequent re-assembly of such subblocks. By partitioning a block into subblocks, a system creates the potential for subblocks to be erroneously added, deleted, rearranged, substituted, duplicated, or in some other way exposed to a greater risk of accidental error.

This risk can be reduced or eliminated by calculating the hash (preferably a cryptographic hash) of the block before it is partitioned into subblocks, storing the hash with an entity associated with the block as a whole and then later comparing the stored hash with a computed hash of the reconstructed subblock. Such a check would provide a very strong safety net that would virtually eliminate the risk of undetected errors arising from the use of this invention.

Choosing A Partitioning Function

Although the requirements for the block partitioning function F are not stringent, care should be taken to select a function that suits the application to which it is to be applied.

In situations where the data is highly structured and knowledge of the data is available, a choice of an F that tends to place subblock boundaries at positions in the data that correspond to obvious boundaries in the data could be advantageous. However, in general, F should be chosen from the class of narrow hash functions. Use of a narrow hash function for F provides both efficiency and a (deterministic) randomness that will enable the implementation to operate effectively over a wide-range of data.

One of the most important properties of F is the probability that F will place a boundary at any particular point when applied to completely random data. For example, a function with a probability of one would produce a boundary between each bit (or byte), whereas a function with a probability of zero would never produce any boundaries at all. In a real application, a more moderate probability would be chosen (e.g. 1/1024) so as to yield useful subblock sizes. The probability can be tuned to suit the application.

We end this section with an example of a narrow hash function that has been implemented and tested and seems to perform well on a variety of data types. The hash function calculates a hash value from three bytes.

$$H(b_1, b_2, b_3) = ((40543 \times ((b_1 \ll 8) \oplus (b_2 \ll 4) \oplus b_3)) \gg 4) \quad | \quad \rho$$

The following notation has been used. "$\times$" is multiplication. "$\ll$" is left bit shift. "$\gg$" is right bit shift. "$\oplus$" is exclusive or. "$|$" is modulo. The constant ρ is

the inverse of the probability of placing a boundary at an arbitrary position in a randomly generated block of data, and can be set to any integer value in [0.65535]. However, in practice it seems to be advantageous to choose values that are prime (Mersenne primes seem to work well). The value 40543 was chosen carefully in accordance with the guidelines provided in pages 508-513 of the book:

> Knuth D.E., "The Art of Computer Programming: Volume 3: Sorting and Searching", Addison Wesley, 1973.

The function evaluates to a value in the range $[0, \rho - 1]$ and can be used in practice by placing a boundary at each point where the preceding three bytes hash to a predetermined constant value V. This would imply that its arguments $b_1 \ldots b_3$ correspond to the argument A in aspect one above. To avoid pathological behaviour in the commonly occurring case of runs of zeros, it is wise to choose a non-zero value for V.

In a real implementation, ρ was set to 511 and V was set to one.

Although early aspects of the detailed description of the invention refer to a function F that places a boundary when its output value falls within a predetermined subclass of the set of possible output values, it should be noted that the combination of F and its use in the invention can always be viewed as equivalent to a boolean function $B(x, y, z)$ (where x and y are blocks of data) where

$$B(x, y, z) = G(F(x, y, z))$$

and where G is a function accepting a value of whatever type F returns and returning a boolean, being *true* iff the value from F falls within a predetermined subclass defined for F. The z arguments have been included to indicate that the functions could depend on other information too.

Placing An Upper And Lower Bound On The Subblock Size

The use of data-dependent subblock boundaries provides a way to deterministically partition similar portions of data in a context-independent way. However, if artificial bounds are not placed on the subblock size, particular kinds of data will yield subblocks that are either too large or too small to be effective. For example, if a file contains a block of a million identical bytes, any deterministic function F (that operates at the byte level) must either partition the block into one subblock or a million subblocks. Both alternatives are undesirable.

A solution to this problem is to artificially impose an upper bound U and a lower bound L on the subblock size. There seem to be a limitless number of ways of doing this. Here are some examples:

Upper bound: Subdivide subblocks defined by F that are longer than U bytes at the points, U, $2U$, $3U$, and so on, where U is the chosen upperbound on subblock size.

Upper bound: Subdivide subblocks that are longer than U bytes at points determined by a secondary hash function.

Lower bound: Of the set of boundaries that bound subblocks less than L bytes long, remove those boundaries that are closer to their neighbouring boundaries than their neighbouring boundaries are to their neighbouring boundaries.

Lower bound: If the block is being scanned sequentially, do not place a boundary unless at least L bytes have been scanned since the previous boundary.

Lower bound: Of the set of boundaries that bound subblocks less than L bytes long, remove those boundaries that satisfy some secondary hash function.

Lower bound: Of the set of boundaries that bound subblocks less than L bytes long, remove randomly chosen boundaries until all the resulting subblocks are at least L bytes long.

Many other such schemes could be devised.

Partitioning Blocks Using Non Data-Dependent Means

Although this invention will usually be applied using data-dependent, variable-length subblocks, it can also be applied using non data-dependent subblocks. For example, the input blocks could simply be partitioned into n-byte blocks. Non data-dependent partitionings could be very effectively applied to blocks whose content varies but does not move about (i.e. the bits or bytes of the data are modified, but bits or bytes are not inserted, deleted, or re-arranged).

Another way of using fixed-length blocks would be to use many different overlapping partitionings of fixed-length blocks.

There are an infinite number of other ways of partitioning a block into subblocks without referring to its content.

The Use Of Multiple Partitionings

In most applications the use of just one partitioning into subblocks will be sufficient. However, in some applications there may be a need for more than one subblock partitioning. For example, in applications where channel space is expensive, it may be appropriate to partition each block of data in W different ways using W different functions $F_1 \ldots F_W$ where each function provides a different average subblock size. For example, four different partitions could be performed using functions that provide subblocks of average length 256 bytes, 1K, 10K, and 100K. By providing a range of different sizes of subblocks to choose from, such an organization could

simultaneously indicate large blocks extremely efficiently, while still retaining fine-grained subblocks so that minor changes to the data do not result in voluminous updates (Figure 8).

The efficiency of such a scheme could be improved by performing the partitioning all in one operation using increasing constraints on a single F. For example, the example hash function that was described earlier could be used, but with different values of the constant ρ being used to determine the different levels of subdivision. By choosing appropriately related values of ρ, the set of boundaries that could be produced by the different F could be arranged to be subsets of each other, resulting in a tree structure of subblocks. For example, values of ρ of 32, 64, and 128, and 256 could be used. Figure 9 shows how the subblocks of four levels of the tree could relate to each other:

A further method could define the hash of a larger block to be the hash of the hashes of its component blocks.

Multiple partitionings may also be useful simply to provide a wider pool of subblocks to recognize. For example, it may be appropriate to partition each block of data in W different ways using W different functions $F_1 \ldots F_W$ where each function yields roughly the same subblock sizes, but at different positions in the block.

Another technique would be to create an additional set of boundaries based on the boundaries provided by a hash function. For example, a fractal algorithm could be used to partition a block based upon some other partitioning provided by a function F.

Comparing Subblocks

In most applications of this invention, there will be a need at some stage to identify identical subblocks. This can be done in a number of ways:

- Compare the subblocks themselves.
- Compare the hashes of the subblocks.
- Compare identities of the subblocks.
- Compare references to the subblocks.

In most cases, the problem reduces to that of taking a group of subblocks of data and finding all subsets of identical subblocks. This is a well-solved problem and discussion of various solutions can be found in the following books:

Knuth D.E., "The Art of Computer Programming: Volume 1: Fundamental Algorithms", Addison Wesley, 1973.

Knuth D.E., "The Art of Computer Programming: Volume 3: Sorting and Searching", Addison Wesley, 1973.

In most cases, the problem is best solved by creating a data structure that maintains the subblocks, or references to the subblocks, in sorted order, and then inserting each subblock one at a time into the data structure. Not only does this identify all currently identical subblocks, but it also establishes a structure that can be used to determine quickly whether incoming subblocks are identical to any of those already held. The following data structures are described in the books referenced above and provide just a sample of the structures that could be used:

- Hash tables.
- Sorted trees (binary, N-ary, AVL).
- Sorted linked lists.
- Sorted arrays.

Of the multitude of solutions to the problem of matching blocks of data, one solution is worthy of special attention: the hash table. Hash tables consist of a (usually) finite array of slots into which values may be inserted. To add a value to a hash table, the value is hashed (using a hash function that is usually selected from the class of narrow hash functions) into a slot number and the value is inserted into that slot. Later, the value can be retrieved in the same manner. Provisions must be made for the case where two data values to be stored in the same table hash to the same slot number.

Hash tables are likely to be of particular value in the implementation of this invention because:

- They provide very fast (essentially constant time) access.
- Many applications will need to calculate a strong one-way hash of each subblock anyway, and a portion of this value can be used to index the hash table.

Particularly effective would be a hash table indexed by a portion of a strong one-way hash of the subblocks it stores, with each table entry containing (a) the strong one-way hash of the subblock, and (b) a pointer to the subblock stored elsewhere in memory.

The Use Of Compression, Encryption, and Integrity Techniques

Various aspects of the invention could be enhanced by the use of data compression, data encryption, and data integrity techniques. The applications of these techniques include, but are not limited to the following applications:

- Any subblock that is transmitted or represented in its raw form could alternatively be transmitted or represented in a compressed or encrypted form.

- Subblocks could be compressed and encrypted before further processing by aspects of this invention.
- Blocks could be compressed and encrypted before further processing by aspects of this invention.
- Communications or representations could be compressed or encrypted.
- Any component could carry additional checking information such as checksums or digests of the data in the component.
- Ad-hoc data compression techniques could be used to further compress references and identities or consecutive runs of references and identities.

Storage Of Variable-Length Subblocks On Disk

The division of data into subblocks of varying length presents some storage organization problems if the subblocks are to be stored independently of each other, as most hardware disk systems are organized to store an array of fixed-length blocks (e.g. one million 512-byte blocks) rather than variable-length ones. Here are some techniques that could be used to tackle this problem:

- Each subblock could be stored in an integral number of disk blocks, with some part of the last disk block being wasted. For randomly sized subblocks, this scheme will waste on average half a disk block per subblock.
- Create a small subset of different bucket sizes (e.g. powers of two) and create arrays on the disk that pack collections of these buckets efficiently into the disk blocks. For example, if disk blocks were 512 bytes long, one could fairly efficiently pack five 200-byte buckets into an array of two disk blocks. Each subblock would be stored in the smallest bucket size that would hold the subblock, with the unused part of the bucket being wasted.

- Treat the disk blocks as a vast array of bytes and use well-established heap management techniques to manage the array. A sample of such techniques appears in pages 435-451 of the book:

  Knuth D.E., "The Art of Computer Programming: Volume 1: Fundamental Algorithms", Addison Wesley, 1973.

The Use Of Concurrency

Two processes are said to be concurrent if their execution takes place in some sense at the same time:

- In interleaving concurrency, some or all of the operations performed by the two processes are interleaved in time, but the two processes are never both executing at exactly the same instant.
- In genuine concurrency, some or all of the operations performed by the two processes are genuinely executed at the same instant.

Implementations of the present invention could incorporate either form of concurrency to various degrees. In most of the aspects described earlier, some subset of the steps of each aspect could be performed concurrently. In particular (without limitation):

- A block could be split into parts and each part partitioned concurrently.
- The processing of subblocks defined during a sequential partitioning of a block need not be deferred until the entire block has been partitioned. In particular, the hashes of already-defined subblocks could be calculated and compared while further subblocks are being defined.

- Communicating entities implementing aspects that decompose and compose blocks could execute concurrently.
- Where more than one block must be partitioned for processing, such partitioning can occur concurrently.

Many more forms of concurrency within aspects of this invention could be identified.

Example: Partitioning A Block

We now present a simple example of how a block might be partitioned in practice. Consider the following block of bytes:

$$b_1 \quad b_2 \quad b_3 \quad b_4 \quad b_5 \quad b_6 \quad b_7 \quad b_8 \quad b_9 \ldots$$

In this example, the example hash function H will be used to partition the block and boundaries will be represented by pairs such as b_6/b_7. We will assume that H returns a boolean value based on its argument and that a boundary is to be placed at each $b_i/b_i + 1$ for which $F(b_i - 2, b_i - 1, b_i)$ evaluates to *true*.

As the hash function accepts 3 byte arguments, we start at b_3/b_4 and evaluate $H(b_1.b_2.b_3)$. This turns out to be *false*, so we move to b_4/b_5 and evaluate $H(b_2.b_3.b_4)$. This turns out to be *true* so a boundary is placed at b_4/b_5. Next, we move to b_5/b_6 and evaluate $H(b_3, b_4, b_5)$. This turns out to be *false* so we move on. $H(b_4.b_5.b_6)$ is *true* so we place a boundary at b_6/b_7. This process continues until the end of the block is reached.

$$b_1 \quad b_2 \quad b_3 \quad b_4 \quad | \quad b_5 \quad b_6 \quad | \quad b_7 \quad b_8 \quad b_9 \ldots$$

Some variations on this approach are:

- Imposition of a lower bound L on subblock size by skipping ahead L bytes following the placement of each boundary.
- Imposition of an upper bound U on block size by artificially placing a boundary if U bytes have been processed since the last boundary was placed.
- Improving the efficiency of the hash calculations by using some part of the calculation of the hash of the bytes at one position to calculate the hash at the next position. For example, it may be more efficient to calculate $H(x, y, z)$ if $H(*, x, y)$ has already been calculated. For example, the Internet IP checksum is organized so that a single running checksum value can be maintained, with bytes entering the window being added to the checksum and bytes exiting the window being subtracted from the checksum.
- Applying this algorithm in reverse starting from the end of the block and working backwards.
- Establishing the subblock enclosing a particular point (chosen from anywhere within the block) by exploring in both directions from the point looking for the nearest boundary in each direction.
- Finding all subblock boundaries in one step by evaluating F for all positions in parallel.

Example: Forming A Table of Hashes

Once a block has been partitioned, the hash of each subblock can be calculated to form a table of hashes (Figure 24).

This table of hashes can be used to determine if a new subblock is identical to any of the subblocks whose hashes are in the table. To do this the new subblock's hash is calculated and a check made to see if the hash is in the table.

In Figure 24. the table of hashes looks like an array of hashes. However, the table of hashes could be stored in a wide variety of data structures (e.g. hash tables. binary trees).

Example Application: A File Comparison Utility

As the invention provides a new way of finding similarities between large volumes of data, it follows that it should find some application in the comparison of data.

In one aspect, the invention could be used to determine the broad similarities between two files being compared by a file comparison utility. The utility would partition each of the two files into subblocks, organize the hashes of the subblocks somehow (e.g. using a hash table) to identify all identical subblocks, and then use this information as a framework for reporting similarities and differences between the two files.

In a similar aspect, the invention could be used to find similarities between the contents of large numbers of files in a file system. A utility incorporating the invention could read each file in an entire file system, partition each into subblocks and then insert the subblocks (or hashes of the subblocks) into one huge table (e.g. implemented by a hash table or a binary tree). If each entry in the table carried the name of the file containing it as well as the position of the subblock within the file, the table could later be used to identify those files containing identical portions of data.

If, in addition, a facility was added for recording and comparing the hashes of the entire contents of files and directory trees, a utility could be constructed that could identify all largely similar structures within a file system. Such a utility would be immensely useful when (say) attempting to merge the data on several similar backup tapes.

Example Application: A Fine-Grained Incremental Backup System

In a fine-grained incremental backup system, two entities $E1$ and $E2$ (e.g. two computers on a network) wish to repeatedly backup a file X at $E1$ such that the old version of the file Y at $E2$ will be updated to become a copy of the new version of the file X at $E1$ (without modifying X). The system could work as follows:

Each time $E1$ performs a backup operation, it partitions X into subblocks and writes the hashes of the subblocks to a shadow file S. It might also write a hash of the entire contents of X to the shadow file. After the backup has been completed, X will be the same as Y and so the shadow file S will correspond to both X and Y. Once X is again modified (during the normal operation of the computer system), S will correspond only to Y. S is used during the next backup operation.

To perform the backup, $E1$ compares the hash of Y (stored in S) against the hash of X to see if X has changed (it could also use the modification date file attribute of the file). If X hasn't changed, there is no need to perform any further backup action. If X has changed, $E1$ partitions X into subblocks and compares the hashes of these subblocks with the hashes in the shadow file S, so as to find all identical hashes. Identical hashes identify identical subblocks in Y that can be transmitted by reference. $E1$ then transmits the file as a mixture of raw subblocks and references to subblocks whose hashes appear in S and which are therefore known to appear as subblocks in Y. $E1$ can also transmit references to subblocks already transmitted. References can take many forms including (without limitation):

- A hash of the subblock.
- The number of the subblock in the list of subblocks in Y.
- The number of a subblock previously transmitted.

- A range of any of the above.

Throughout this process $E1$ can be constructing the new shadow file corresponding to X. Figure 25 illustrates the backup process.

To reconstruct X from Y and the incremental backup information being sent from $E1$, $E2$ partitions Y into subblocks and calculates the hashes of the subblocks (It could do this in advance during the previous backup). It then processes the incremental backup information, copying subblocks that were transmitted raw and looking up the references either in Y or in the part of X already reconstructed.

Because information need only flow from $E1$ to $E2$ during the backup operation, there is no need for $E1$ and $E2$ to perform the backup operation concurrently. $E1$ can perform its side of the backup operation in isolation, producing an incremental backup file that can be later processed by $E2$.

There is a tradeoff between 1) the approximate ratio between the size of each file and that of its shadow, and 2) the mean subblock size. The higher the mean subblock size (as determined by the partitioning method used (including F)), the fewer subblocks per unit file length, and hence the shorter the shadow size per unit file length. However, increasing mean subblock sizes implies increasing the granularity of backups which can cause an increase in the size of the incremental backup file. There is also a tradeoff between the shadow file size and the hash width. A shadow file that uses 128-bit hashes will be about twice as long as one that uses 64-bit hashes. All these tradeoffs must be considered closely when constructing an implementation.

In a real existing implementation of this backup scheme, the exact format of the shadow file S is:

Bytes	Description
16	MD5 digest of the file Y corresponding to this shadow file.
16	MD5 digest of the first subblock in Y.
16	MD5 digest of the second subblock in Y.
..	...
16	MD5 digest of the last subblock in Y.
16	MD5 digest of the rest of this shadow file.

The first field contains the MD5 digest (a form of cryptographic hash) of the entire contents of Y. This is included so that it can be copied to the incremental backup file so as to provide a check later that the incremental backup file is not being applied to the wrong version of Y. It could also be used to determine if *any* change has been made to X since the previous backup Y was taken. The first field is followed by a list of the MD5 digests of the subblocks in Y in the order in which they appear in Y. Finally, a digest of the contents of the shadow file (less this field) is included at the end so as to enable the detection of any corruption of the shadow file.

The format of the incremental backup file is as follows:

Bytes	Description
16	MD5 digest of Y.
16	MD5 digest of X.
..	Zero or more ITEMS.
16	MD5 digest of the rest of the incremental backup file.

The first two fields of the incremental backup file contain the MD5 digest of the old and new versions of the file. The hash of the new version X is calculated directly

from X. The hash of the old version is obtained from the first field of the shadow file. These two values enable the remote backup entity $E2$ to check that:

- The backup file Y (to be updated) is identical to the one from which the shadow file was generated.
- The reconstructed X is identical to the original X.

The two checking fields are followed by a list of items followed by a checking digest of the rest of the incremental backup file.

Each item in the list of items describes one or more subblocks in the list of subblocks that can be considered to constitute X. There are three kinds of item, and so each item commences with a byte having a value one, two, or three to indicate the kind of item. Here is a description of the content of each of the three kinds of item:

1. The 32-bit index of a subblock in Y. Because $E2$ possesses Y, it can partition Y itself to construct the same partitioning that was used to create the shadow file. Thus $E1$ doesn't need to send the hash of any subblock that is in both X and Y. Instead, it need only send the *index* of the subblock in the list of subblocks constituting Y. This list is represented by the list of hashes in S. As 32-bits is wide enough for an index in practice, the saving gained by communicating a 32-bit index instead of a hash is 98 bits for each such item.
2. A pair of 32-bit numbers being the index of the first and last subblock of a range of subblocks in Y. Old and new versions of files often share large contiguous ranges of subblocks. The use of this kind of item allows such ranges to be represented using just 64 bits instead of a long run of instances of the first kind of item.

3. A 32-bit value containing the number of bytes in the subblock, followed by the raw content of the subblock. This kind of item is used if the subblock to be transmitted is not present in Y.

In the implementation, all the values are coded in little-endian form. Big-endian could be used equally as well.

The existing implementation could be further optimized by (without limitation):

- Adding an additional kind of item that refers to subblocks in X already transmitted:
- Adding an additional kind of item that refers to ranges of subblocks in X already transmitted;
- Employing data compression techniques to compress the raw blocks in the third kind of item.
- Using the first hash in the shadow file to check to see if the entire file has changed at all before performing the backup process described above.
- Replacing hashes in S of subblocks in Y by references to other hashes in S (where the hashes (and hence subblocks) are identical). Repeated runs of hashes could also be replaced by pointers to ranges of hashes.

The scheme described above has been described in terms of a single file. However, the technique could be applied repeatedly to each of the files in a file system, thus providing a way to back up an entire file system. The shadow information for each file in the file system could be stored inside a separate shadow file for each file, or in a master shadow file containing the shadows for one or more (or all) files in the file system.

Although most redundancy in a file system is likely to be found within different versions of each file, there may be great similarities between versions of different files. For example, if a file is renamed, the "new" file will be identical to the "old" file. Such redundancy can be catered for by comparing the hashes of all the files in the old and new versions of a file system. In addition, similarities between different parts of different files can be exploited by comparing the hashes of subblocks of each file to be backed up against the hashes of the subblocks of the entire old version of the file system.

If $E2$ has lots of space, a further improvement could be for $E1$ to retain the shadows of all the previous versions of the file system, and for $E2$ to retain copies of all the previous versions of the file system. $E1$ could then refer to every block it has ever seen. This technique could also be applied on a file-by-file basis.

In a further variant, the dependence on the ordering of subblocks could be abandoned and $E1$ could simply keep a shadow file containing a list of the hashes of all the subblocks in the previous version (or versions) of the file or file system. $E2$ would then need to record only a single copy of each unique subblock it has ever received from $E1$.

Aspects of the backup application described in this section can be integrated cleanly into existing backup architectures by deploying the new mechanisms within the framework of the existing ones. For example, the traditional methods for determining if a file has changed since the last backup (modification date, backup date and so on) can be used to see if a file needs to be backed up at all, before applying the new mechanisms.

Example Application: A Low-Redundancy File System

We now present an example of a low-redundancy file system that attempts to avoid

storing different instances of the same data more than once. In this example, the file system is organized as shown in Figure 26.

The bottom layer consists of a collection of unique subblocks of varying length that are stored somewhere on the disk. The middle layer consists of a hash table containing one entry for each subblock. Each entry consists of a cryptographic hash of the subblock, a reference count for the subblock, and a pointer to the subblock on disk. The hash table is indexed by some part of the cryptographic hash (e.g. the bottom 16 bits). Although a hash table is used in this example, many other data structures (e.g. a binary tree) could also be used to map cryptographic hashes to subblock entries . It would also be possible to index the subblocks directly without the use of cryptographic hashes.

The top layer consists of a table of files that binds filenames to lists of subblocks, each list being a list of indexes into the hash table. Each hash table entry corresponds to a single unique subblock (except possibly in the case of overflow) and contains the cryptographic hash of the subblock along with a reference count and a pointer to the subblock on the disk. The reference count records the number of references to the subblock that appear in the entire set of files in the file table. The issue of hash table "overflow" can be addressed using a variety of well-known overflow techniques such as that of attaching a linked list to each hash slot.

When a file is read, the list of hash table indexes is converted to pointers to subblocks of data using the hash table. If random access to the file is required, extra information about the length of the subblocks could be added to the file table and/or hash table so as to speed access.

Writing a file is more complicated. During a sequential write, the data being written is buffered until a subblock-boundary is reached (as determined by whatever boundary function is being used). The cryptographic hash of the new subblock is then calculated and used to look up the hash table. If the subblock is unique (i.e. there

is no entry for the cryptographic hash), it is added to the data blocks on the disk and an entry is added to the hash table. A new subblock number is added to the list of blocks in the file table. If, on the other hand, the subblock already exists, the subblock need not be written to disk. Instead, the reference count of the already-existing subblock is incremented, and the subblock's hash table index is added to the list of blocks in the file's entry in the file table.

Random access writes are more involved, but essentially the same principles apply.

If a record were kept of subblocks created since the last backup, backing up this file system could be very efficient indeed.

One enhancement that could be made is to exploit unused disk space. Instead of automatically ignoring or overwriting subblocks whose reference count has dropped to zero, the low-redundancy file system could move them to a pool of unused subblocks. These subblocks, while not present in any file, could still form part of the subblock pool referred to when checking to see if incoming subblocks are already present in the file system. The space consumed by subblocks in the unused subblock pool would be recycled only when the disk was full. In the steady state, the "unused" portion of the disk would be filled by subblocks in the unused subblock pool.

Although this section has specifically described a low-redundancy file system, this aspect of the invention is really a general purpose storage system that could be applied at many levels and in many roles in information processing systems. For example:

- The technique could be used to implement a low-redundancy virtual memory system. The contents of memory could be organized as a collection of subblocks.

- The technique could be used to increase the efficiency of an on-chip cache.

Example Application: A Communication System

We now present a method for reducing duplicate transmissions in communications systems. Consider two entities $E1$ and $E2$, where $E1$ must transfer a block of data X to $E2$. $E1$ and $E2$ need never have communicated previously with each other.

The conventional way to perform the transmission is simply for $E1$ to transmit X to $E2$. However, here, $E1$ first partitions X into subblocks and calculates the hash of each subblock using a hash function. It then transmits the hashes to $E2$. $E2$ then looks up the hashes in a table of hashes of all the subblocks it already possesses. $E2$ then transmits to $E1$ information (e.g. a list of subblock numbers) identifying the subblocks in X that $E2$ does not already possess. $E1$ then transmits just those subblocks.

Another way to perform the transaction would be for $E2$ to first transmit to $E1$ the hashes of all the subblocks it possesses (or perhaps a well chosen subset of them). $E1$ could then transmit references to subblocks in X already known to $E2$ and the actual contents of subblocks in X not known to $E2$. This scheme could be more efficient than the earlier scheme in cases where $E2$ possesses less subblocks than there are in X.

Another way to perform the transaction is for $E1$ and $E2$ to conduct a more complicated conversation to establish which subblocks $E2$ possesses. For example. $E2$ could send $E1$ the hashes of just some of the subblocks it possesses (perhaps the most popular ones). $E1$ could then send to $E2$ the hashes of other subblocks in X. $E2$ could then reply indicating which of those subblocks it truly does not possess. $E1$ could then send to $E2$ the subblocks in X not possessed by $E2$.

In a more sophisticated system, $E1$ and $E2$ could keep track of the hashes of the subblocks possessed by the other. If either entity ever sent (for whatever reason) a reference to a subblock not possessed by the other entity, the latter entity could simply send back a request for the subblock to be transmitted explicitly and the former entity could send the requested subblock.

The communication application described above considers the case of just two communicants. However, there is no reason why the scheme could not be generalized to cover more than two communicants communicating with each other in private and in public (using broadcasts). For example, to broadcast a block, a computer C_1 could broadcast a list of the hashes of the block's subblocks. Computers $C_2 \ldots C_N$ could then each reply indicating which subblocks they do not already possess. C_1 could then broadcast subblocks that many of the other computers do not possess, and send the subblocks missing from only a few computers to those computers privately.

All these techniques have the potential to greatly reduce the amount of information transmitted between computers.

These techniques would be very efficient if they were implemented on top of the file system described earlier, as the file system would already have performed the work of organizing all the data it possesses into indexed subblocks. The potential savings in communication that could be made if many different computer systems shared the same subblock partitioning algorithm suggests that some form of universal standardization on a particular partitioning method would be a worthy goal.

Example Application: A Subblock Server

Aspects of the invention could be used to establish a subblock server on a network so as to reduce network traffic. A subblock server could be located in a busy part of a network. It would consist of a computer that breaks each block of data it

sees into subblocks, hashes the subblocks, and then stores them for future reference. Other computers on the network could send requests to the server for subblocks, the requests consisting of the hashes of subblocks the server might possess. The server would respond to each hash, returning either the subblock corresponding to the hash, or a message stating that the server does not possess a subblock corresponding to the hash.

Such a subblock server could be useful for localizing network traffic on the Internet. For example, if a subnetwork (even a large one for (say) an entire country) placed a subblock server on each of its major Internet connections, then (with the appropriate modification of various protocols) much of the traffic into the network could be eliminated. For example, if a user requested a file from a remote host on another network, the user's computer might issue the request and receive, in reply, not the file, but the hashes of the file's subblocks. The user's computer could then send the hashes to the local subblock server to see if the subblocks are present there. It would receive the subblocks that are present and then forward a request for the remaining subblocks to the remote host. The subblock server might notice the new subblocks flowing through it and archive them for future reference. The entire effect would be to eliminate most repeated data transfers between the subnetwork and the rest of the Internet. However, the security implications of schemes such as these would need to be closely investigated before there were deployed.

A further step could be to create "virtual" subblock servers that store the hashes of subblocks and their location on the Internet rather than the subblocks and their hashes.

CLAIMS

Note: Claims 1 to 11 will be referred to as the **partitioning claims**.

1. A method for partitioning a block b into one or more subblocks, the method using the component:

(i) a deterministic or non-deterministic function F that returns one of at least two values, and whose arguments include at least a block of A bits and a block of B bits, where A and B are natural numbers;

and comprising the step of:

a. Basing the positions of subblock boundaries on the positions k in the block for which $F(b_{k-A} \ldots b_k, b_{k+1} \ldots b_{k+B})$ falls within a predetermined subclass of the set of possible function result values.

2. A method according to claim 1, for locating the nearest subblock boundary on a particular side of a particular position p within a block, but replacing step (a) with:

a. Evaluating $F(b_{p-A} \ldots b_p, b_{p+1} \ldots b_{p+B})$ for increasing (or decreasing) p until the result of F falls within a predetermined subclass of the set of possible function result values, the position of the resultant boundary being based on this position.

3. A method according to any claim above, for partitioning a block into one or more subblocks, wherein boundaries may be added and removed in accordance with a further method.

4. A method according to any claim above, for partitioning a block into one or more subblocks, wherein an upperbound U on the subblock size is imposed.

5. A method according to any claim above, for partitioning a block into one or more subblocks, wherein a lowerbound L on the subblock size is imposed.

6. A method according to any claim above, for partitioning a block into one or more subblocks, wherein an upperbound U on the subblock size is imposed and a lowerbound L on the subblock size is also imposed.

7. A method according to one or more of the claims above, wherein more than one partitioning function (e.g. F_1, F_2, ...) and method are applied independently to the block b so as to form more than one group of subblocks.

8. A method for partitioning a block into one or more subblocks by dividing the block into subblocks of equal size.

9. A method for partitioning a block into one or more subblocks by dividing the block into subblocks of a small number of different sizes.

10. A method according to one of the claims above, wherein additional subblocks are formed from one or more groups of subblocks.

11. A method according to one of the claims above, wherein an additional hierarchy of subblocks is formed from one or more contiguous groups of subblocks.

12. A method in accordance with any partitioning claim, for partitioning a block into subblocks and forming a corresponding collection of hashes, comprising the steps of:

a. Partitioning the block into one or more subblocks in accordance with any partitioning claim;

b. Calculating the hash of one or more subblocks using a hash function H.

13. A method in accordance with any partitioning claim, for constructing a projection of a block, comprising the steps of:

a. Partitioning the block into one or more subblocks in accordance with any partitioning claim;

b. Forming a projection which is an ordered or unordered list containing identities (e.g. subblocks or hashes of subblocks) of, or references to, one or more of the subblocks.

14. A method, in accordance with any claim above, for finding identical portions within a group of one or more blocks comprising the steps of:

a. Partitioning one or more of said blocks into one or more subblocks in accordance with any claim above:

b. Comparing the subblocks or the identities (e.g. hashes) of the subblocks.

15. A method in accordance with any partitioning claim, for representing one or more blocks, involving the following components:

(i) A method for storing and retrieving subblocks;

(ii) A mapping from block representatives (e.g. filenames) to lists of entries that identify subblocks;

whereby the modification of data in a stored block involves the following steps:

a. Partitioning the new data into subblocks in accordance with any partitioning claim;

b. Adding subblocks in the new data that are not already in the collection of stored subblocks to the collection of stored subblocks, and updating the subblock list associated with the block being modified;

16. A method for an entity $E1$ to communicate a group X of one or more subblocks $X_1 \ldots X_n$ to $E2$ where $E1$ possesses the knowledge that $E2$ possesses a group Y of zero or more subblocks $Y_1 \ldots Y_m$ comprising the following step:

a. Transmitting from $E1$ to $E2$ the contents of a subset of zero or more subblocks in X, and the remaining subblocks as references which may take (but are not limited to) the following forms:

(i) a hash of a subblock;

(ii) a reference to a subblock in Y;

(iii) a reference to a range of subblocks in Y;

(iv) a reference to a subblock already transmitted;

(v) a reference to a range of subblocks already transmitted.

17. A method in accordance with claim 16 and any partitioning claim, for an entity $E1$ to communicate a block X to $E2$ where $E1$ possesses the knowledge that $E2$ possesses a group Y of subblocks $Y_1 \ldots Y_m$ comprising step (a) of claim 16 preceded by the step:

s. Partitioning X into subblocks $X_1 \ldots X_n$ in accordance with any partitioning claim.

18. A method in accordance with claim 16 and any partitioning claim, for an entity $E1$ to communicate one or more subblocks of a group X of subblocks $X_1 \ldots X_n$ to $E2$ where $E1$ possesses the knowledge that $E2$ possesses the block Y, comprising step (a) of claim 16 preceded by the step:

s. Partitioning Y into subblocks $Y_1 \ldots Y_m$ in accordance with any partitioning claim.

19. A method in accordance with claim 16 and any partitioning claim, for an entity $E1$ to communicate a block X to $E2$ where $E1$ possesses the knowledge that $E2$ possesses block Y, comprising step (a) of claim 16 preceded by the steps:

s1. Partitioning X into subblocks $X_1 \ldots X_n$ in accordance with any partitioning claim.

s2. Partitioning Y into subblocks $Y_1 \ldots Y_m$ in accordance with any partitioning claim.

20. A method for constructing a block D from a group X of one or more subblocks $X_1 \ldots X_n$ and a group Y of zero or more subblocks $Y_1 \ldots Y_m$ such that X can be constructed from Y and D, comprising the step:

a. Constructing D from at least one of the following components:

(1) the contents of one or more subblocks in X;

(2) references to subblocks in Y or to subblocks included in D, or to a range of subblocks from either D or Y.

21. A method in accordance with claim 20 and any partitioning claim, constructing a block D from a block X and a group Y of subblocks $Y_1 \ldots Y_m$ such that X can be constructed from Y and D, comprising step (a) of claim 20 preceded by the step:

s. Partitioning X into subblocks $X_1 \ldots X_n$ in accordance with any partitioning claim.

22. A method in accordance with claim 20 and any partitioning claim, constructing a block D from a group X of subblocks $X_1 \ldots X_n$ and a block Y such that X can be constructed from Y and D, comprising step (a) of claim 20 preceded by the step:

s. Partitioning Y into subblocks $Y_1 \ldots Y_m$ in accordance with any partitioning claim.

23. A method in accordance with claim 20 and any partitioning claim, constructing a block D from a block X and a block Y such that X can be constructed from Y and D, comprising step (a) of claim 20 preceded by the steps:

s1. Partitioning X into subblocks $X_1 \ldots X_n$ in accordance with any partitioning claim.

s2. Partitioning Y into subblocks $Y_1 \ldots Y_m$ in accordance with any partitioning claim.

24. A method for constructing a block D from a group X of subblocks $X_1 \ldots X_n$ and a projection of a block Y (or a projection of a group Y of subblocks $Y_1 \ldots Y_m$), such that X can be constructed from Y and D, comprising the step:

a. Constructing D from at least one of the following components:

(1) the contents of one or more subblocks in X;

(2) references to subblocks in Y or to subblocks included in D, or to a range of subblocks from either D or Y.

25. A method in accordance with claim 24 and any partitioning claim, constructing a block D from a block X and a projection of Y such that X can be constructed from Y and D, comprising the step of claim 24 with the following step inserted before step (a):

s. Partitioning X into subblocks $X_1 \ldots X_n$ in accordance with any partitioning claim;

26. A method for constructing a block X (or group X of subblocks $X_1 \ldots X_n$) from a group Y of subblocks $Y_1 \ldots Y_m$ and a block D, comprising the step of:

a. Constructing X from D and Y by constructing the subblocks of X based on one or more of:

(i) references in D to subblocks in Y:

(ii) references in D to subblocks in D;

(iii) references in D that specify a range of subblocks in Y;

(iv) references in D that specify a range of subblocks in D;

(v) subblocks contained within D;

(vi) other data elements in D.

27. A method in accordance with claim 26 and any partitioning claim, for constructing a block X (or group X of subblocks $X_1 \ldots X_n$) from a block Y and a block D, comprising the step of claim 26 with the following step inserted before step (a):

s. Partitioning Y into subblocks $Y_1 \ldots Y_m$ in accordance with any partitioning claim.

28. A method for transmitting a group X of subblocks $X_1 \ldots X_n$ from one entity $E1$ to another entity $E2$, comprising the steps of:

a. Transmitting from $E1$ to $E2$ an identity of one or more subblocks;

b. Transmitting from $E2$ to $E1$ information communicating the presence or absence of subblocks at $E2$;

c. $E1$ transmitting to $E2$ at least the subblocks identified in step (b) as not being present at $E2$;

29. A method in accordance with claim 28 and any partitioning claim, for communicating a data block X from one entity $E1$ to another entity $E2$, comprising the steps of claim 28 but with the following step inserted before step (a):

s. Partitioning X into subblocks $X_1 \ldots X_n$ in accordance with any partitioning claim.

30. A method in accordance with claim 13 for comparing the contents of two or more blocks comprising the steps:

a. Constructing a projection of each block as described in claim 13;

b. Comparing the projections of the blocks.

31. A method for transmitting a group X of subblocks $X_1 \ldots X_n$ from one entity $E1$ to another entity $E2$, comprising the steps of:

a. Transmitting from $E2$ to $E1$ information communicating the presence or absence at $E2$ of members of a group Y of subblocks $Y_1 \ldots Y_m$;

b. Transmitting from $E1$ to $E2$ the contents of zero or more subblocks in X, and the remaining subblocks as references which may take (but are not limited to) the following forms:

(i) a hash of a subblock;

(ii) a reference to a subblock in Y;

(iii) a reference to a range of subblocks in Y;

(iv) a reference to a subblock already transmitted;

(v) a reference to a range of subblocks already transmitted.

32. A method in accordance with claim 31 and any partitioning claim, for transmitting a block X from one entity $E1$ to another entity $E2$, comprising the steps of claim 31 with the following step inserted before step (a):

s. $E1$ partitioning X into subblocks $X_1 \ldots X_n$ in accordance with any partitioning claim:

33. A method for an entity $E2$ to communicate to an entity $E1$ the fact that $E2$ possesses a group Y of subblocks $Y_1 \ldots Y_m$, comprising the step of:

a. $E2$ transmitting to $E1$ identities or references of the subblocks $Y_1 \ldots Y_m$.

34. A method in accordance with claim 33 and any partitioning claim, for an entity $E2$ to communicate to an entity $E1$ the fact that $E2$ possesses a block Y, comprising the step of claim 33 with the following step inserted before step (a):

s. $E2$ partitioning Y into subblocks $Y_1 \ldots Y_m$ in accordance with any partitioning claim;

35. A method for an entity $E1$ to communicate a subblock X_i to an entity $E2$, comprising the steps:

a. $E2$ sending $E1$ an identity of X_i.

b. $E1$ sending X_i to $E2$.

36. A method in accordance with claim 35 and any partitioning claim, for an entity $E1$ to communicate a subblock X_i to an entity $E2$, comprising the steps of claim 35 with the following step inserted before step (a):

s. $E1$ partitioning a block X into subblocks $X_1 \ldots X_n$ in accordance with any partitioning claim;

37. A method in accordance with any claim above, wherein one or more of the comparisons of subblocks are performed by comparing the hashes of the subblocks, using hashes already available (e.g. as a byproduct of other steps), or calculated for the purpose of performing one or more said comparisons.

38. A method in accordance with any claim above, wherein subsets of identical subblocks within a group of one or more subblocks are identified by inserting each

subblock. an identity of each subblock. a reference of each subblock, or a hash of each subblock, into a data structure.

39. A method in accordance with any claim above. wherein various actions are executed concurrently.

40. An apparatus for partitioning a block b into one or more subblocks, the apparatus comprising:

(i) means for evaluating a deterministic or non-deterministic function F that returns one of at least two values, and whose arguments include at least a block of A bits and a block of B bits, where A and B are natural numbers:

and comprising the step of

a. Generating a set of partitions of b, basing these upon the positions of subblock boundaries on the positions k in the block for which $F(b_{k-A} \ldots b_k, b_{k+1} \ldots b_{k+B})$ falls within a predetermined subclass of the set of possible function result values.

41. An apparatus, in accordance with claim 40, for locating the nearest subblock boundary on a particular side of a particular position p within a block b, wherein step (a) is replaced with

a. Generating a position within b by evaluating $F(b_{p-A} \ldots b_p, b_{p+1} \ldots b_{p+B})$ for increasing (or decreasing) p until the result of F falls within a predetermined subclass of the set of possible function result values. the position being based on this position.

42. An apparatus for partitioning a block into one or more subblocks comprising

(i) means for dividing a block into subblocks of equal size.

43. An apparatus for partitioning a block into one or more subblocks comprising

(i) means for dividing a block into subblocks of a small number of different sizes.

44. An apparatus $E1$ that can communicate a group X of one or more subblocks $X_1 \ldots X_n$ to an entity $E2$ where $E1$ possesses the knowledge that $E2$ possesses a group Y of zero or more subblocks $Y_1 \ldots Y_m$, the apparatus comprising

(i) means for manipulating subblocks, subblock identities, and subblock references:

and comprising the step

a. Transmitting from $E1$ to $E2$ the contents of a subset of zero or more subblocks in X, and the remaining subblocks as references which may take (but are not limited to) the following forms:

(i) a hash of a subblock:

(ii) a reference to a subblock in Y;

(iii) a reference to a range of subblocks in Y;

(iv) a reference to a subblock already transmitted;

(v) a reference to a range of subblocks already transmitted.

45. An apparatus for constructing a block D from a group X of one or more subblocks $X_1 \ldots X_n$ and a group Y of zero or more subblocks $Y_1 \ldots Y_m$ such that X can be constructed from Y and D, the apparatus comprising:

(i) means for manipulating subblocks, subblock identities and subblock references:

and comprising the step

a. Constructing D from at least one of the following components:

(1) the contents of one or more subblocks in X:

(2) references to subblocks in Y or to subblocks included in D, or to a range of subblocks from either D or Y.

46. An apparatus for constructing a block D from a group X of subblocks $X_1 \ldots X_n$ and a projection of a block Y (or a projection of a group Y of subblocks $Y_1 \ldots Y_m$), such that X can be constructed from Y and D, the apparatus comprising

(i) means for manipulating subblocks, subblock identities and subblock references;

and comprising the step:

a. Constructing D from at least one of the following components:

(1) the contents of one or more subblocks in X;

(2) references to subblocks in Y or to subblocks included in D, or to a range of subblocks from either D or Y.

47. An apparatus for constructing a block X (or group X of subblocks $X_1 \ldots X_n$) from a group Y of subblocks $Y_1 \ldots Y_m$ and a block D, the apparatus comprising

(i) means for manipulating subblocks, subblock identities and subblock references;

and comprising the step of

a. Constructing X from D and Y by constructing the subblocks of X based on one or more of:

(i) references in D to subblocks in Y;

(ii) references in D to subblocks in D;

(iii) references in D that specify a range of subblocks in Y;

(iv) references in D that specify a range of subblocks in D:

(v) subblocks contained within D;

(vi) other data elements in D.

48. A system for transmitting a group X of subblocks $X_1 \ldots X_n$ from one apparatus $E1$ to another apparatus $E2$, each apparatus consisting of

(i) means for manipulating subblocks, subblock identities and subblock references;

and the systems execution comprising the steps of:

a. Transmitting from $E1$ to $E2$ an identity of one or more subblocks:

b. Transmitting from $E2$ to $E1$ information communicating the presence or absence of subblocks at $E2$;

c. $E1$ transmitting to $E2$ at least the subblocks identified in step (b) as not being present at $E2$:

49. A system for transmitting a group X of subblocks $X_1 \ldots X_n$ from one apparatus $E1$ to another apparatus $E2$, each apparatus consisting of

(i) means for manipulating subblocks, subblock identities and subblock references;

and comprising the steps of:

a. Transmitting from $E2$ to $E1$ information communicating the presence or absence at $E2$ of members of a group Y of subblocks $Y_1 \ldots Y_m$:

b. Transmitting from $E1$ to $E2$ the contents of zero or more subblocks in X, and the remaining subblocks as references which may take (but are not limited to) the following forms:

(i) a hash of a subblock;

(ii) a reference to a subblock in Y;

(iii) a reference to a range of subblocks in Y;

(iv) a reference to a subblock already transmitted;

(v) a reference to a range of subblocks already transmitted.

50. A system for an apparatus $E2$ to communicate to an apparatus $E1$ the fact that $E2$ possesses a group Y of subblocks $Y_1 \ldots Y_m$, each apparatus consisting of

(i) means for manipulating subblocks, subblock identities and subblock references;

and comprising the step of:

a. $E2$ transmitting to $E1$ identities or references of the subblocks $Y_1 \ldots Y_m$.

51. A system for an apparatus $E1$ to communicate a subblock X_i to an apparatus $E2$, each apparatus consisting of

(i) means for manipulating subblocks, subblock identities and subblock references;

and comprising the steps:

a. $E2$ sending $E1$ an identity of X_i.

b. $E1$ sending X_i to $E2$.

AMENDED CLAIMS
[received by the International Bureau on 24 July 1996 (24.07.96); original claims 1-51 replaced by amended claims 1-31 (10 pages)]

Note: Claims 1 to 9 will be referred to as the **partitioning claims**.

1. A method for partitioning a block b into one or more subblocks, the method using the component:

(i) a deterministic or non-deterministic function F that returns one of at least two values, and whose arguments include at least a block of A bits and a block of B bits, where A and B are natural numbers;

and comprising the step of:

a. Basing the positions of subblock boundaries on the positions k in the block for which $F(b_{k-A} \ldots b_k, b_{k+1} \ldots b_{k+B})$ falls within a predetermined subclass of the set of possible function result values.

2. A method according to claim 1, for locating the nearest subblock boundary on a particular side of a particular position p within a block, but replacing step (a) with:

a. Evaluating $F(b_{p-A} \ldots b_p, b_{p+1} \ldots b_{p+B})$ for increasing (or decreasing) p until the result of F falls within a predetermined subclass of the set of possible function result values, the position of the resultant boundary being based on this position.

3. A method according to any claim above, for partitioning a block into one or more subblocks, wherein boundaries may be added and removed in accordance with a further method.

4. A method according to any claim above, for partitioning a block into one or more subblocks, wherein an upperbound U on the subblock size is imposed.

5. A method according to any claim above, for partitioning a block into one or more subblocks, wherein a lowerbound L on the subblock size is imposed.

6. A method according to any claim above, for partitioning a block into one or more subblocks, wherein an upperbound U on the subblock size is imposed and a lowerbound L on the subblock size is also imposed.

7. A method according to one or more of the claims above, wherein more than one partitioning function (e.g. F_1, F_2, ...) and method are applied independently to the block b so as to form more than one group of subblocks.

8. A method according to one of the claims above, wherein additional subblocks are formed from one or more groups of subblocks.

9. A method according to one of the claims above, wherein an additional hierarchy of subblocks is formed from one or more contiguous groups of subblocks.

10. A method in accordance with any partitioning claim, for partitioning a block into subblocks and forming a corresponding collection of hashes, comprising the steps of:

a. Partitioning the block into one or more subblocks in accordance with any partitioning claim:

b. Calculating the hash of one or more subblocks using a hash function H.

11. A method in accordance with any partitioning claim, for constructing a projection of a block, comprising the steps of:

a. Partitioning the block into one or more subblocks in accordance with any partitioning claim:

b. Forming a projection which is an ordered or unordered list containing identities (e.g. subblocks or hashes of subblocks) of, or references to, one or more of the subblocks.

12. A method, in accordance with any claim above, for finding identical portions within a group of one or more blocks comprising the steps of:

a. Partitioning one or more of said blocks into one or more subblocks in accordance with any claim above:

b. Comparing the subblocks or the identities (e.g. hashes) of the subblocks.

13. A method in accordance with any partitioning claim, for representing one or more blocks, involving the following components:

(i) A method for storing and retrieving subblocks:

(ii) A mapping from block representatives (e.g. filenames) to lists of entries that identify subblocks;

whereby the modification of data in a stored block involves the following steps:

a. Partitioning the new data into subblocks in accordance with any partitioning claim;

b. Adding subblocks in the new data that are not already in the collection of stored subblocks to the collection of stored subblocks, and updating the subblock list associated with the block being modified;

14. A method in accordance with any partitioning claim, for an entity $E1$ to communicate a block X to $E2$ where $E1$ possesses the knowledge that $E2$ possesses a group Y of subblocks $Y_1 \ldots Y_m$, comprising the following steps:

a. Partitioning X into subblocks $X_1 \ldots X_n$ in accordance with any partitioning claim;

b. Transmitting from $E1$ to $E2$ the contents of a subset of zero or more subblocks in X, and the remaining subblocks as references which may take (but are not limited to) the following forms:

(i) a hash of a subblock;

(ii) a reference to a subblock in Y;

(iii) a reference to a range of subblocks in Y;

(iv) a reference to a subblock already transmitted;

(v) a reference to a range of subblocks already transmitted.

15. A method in accordance with any partitioning claim, for an entity $E1$ to communicate one or more subblocks of a group X of subblocks $X_1 \ldots X_n$ to $E2$ where $E1$ possesses the knowledge that $E2$ possesses the block Y, comprising the following steps:

a. Partitioning Y into subblocks $Y_1 \ldots Y_m$ in accordance with any partitioning claim;

b. Transmitting from $E1$ to $E2$ the contents of a subset of zero or more subblocks in X, and the remaining subblocks as references which may take (but are not limited to) the following forms:

(i) a hash of a subblock;

(ii) a reference to a subblock in Y;

(iii) a reference to a range of subblocks in Y;

(iv) a reference to a subblock already transmitted;

(v) a reference to a range of subblocks already transmitted.

16. A method in accordance with any partitioning claim, for an entity $E1$ to communicate a block X to $E2$ where $E1$ possesses the knowledge that $E2$ possesses block Y, comprising the following steps:

a. Partitioning X into subblocks $X_1 \ldots X_n$ in accordance with any partitioning claim;

b. Partitioning Y into subblocks $Y_1 \ldots Y_m$ in accordance with any partitioning claim;

c. Transmitting from $E1$ to $E2$ the contents of a subset of zero or more subblocks in X, and the remaining subblocks as references which may take (but are not limited to) the following forms:

(i) a hash of a subblock:

(ii) a reference to a subblock in Y;

(iii) a reference to a range of subblocks in Y;

(iv) a reference to a subblock already transmitted;

(v) a reference to a range of subblocks already transmitted.

17. A method in accordance with any partitioning claim, for constructing a block D from a block X and a group Y of subblocks $Y_1 \ldots Y_m$ such that X can be constructed from Y and D, comprising the following steps:

a. Partitioning X into subblocks $X_1 \ldots X_n$ in accordance with any partitioning claim;

b. Constructing D from at least one of the following components:

(i) the contents of one or more subblocks in X;

(ii) references to subblocks in Y or to subblocks included in D, or to a range of subblocks from either D or Y.

18. A method in accordance with any partitioning claim, for constructing a block D from a group X of subblocks $X_1 \ldots X_n$ and a block Y such that X can be constructed from Y and D, comprising the following steps:

a. Partitioning Y into subblocks $Y_1 \ldots Y_m$ in accordance with any partitioning claim;

b. Constructing D from at least one of the following components:

(i) the contents of one or more subblocks in X;

(ii) references to subblocks in Y or to subblocks included in D, or to a range of subblocks from either D or Y.

19. A method in accordance with any partitioning claim, for constructing a block D from a block X and a block Y such that X can be constructed from Y and D, comprising the following steps:

a. Partitioning X into subblocks $X_1 \ldots X_n$ in accordance with any partitioning claim;

b. Partitioning Y into subblocks $Y_1 \ldots Y_m$ in accordance with any partitioning claim;

c. Constructing D from at least one of the following components:

(i) the contents of one or more subblocks in X;

(ii) references to subblocks in Y or to subblocks included in D, or to a range of subblocks from either D or Y.

20. A method in accordance with any partitioning claim, for constructing a block D from a block X and a projection of Y such that X can be constructed from Y and D, comprising the following steps:

a. Partitioning X into subblocks $X_1 \ldots X_n$ in accordance with any partitioning claim:

b. Constructing D from at least one of the following components:

(i) the contents of one or more subblocks in X;

(ii) references to subblocks in Y or to subblocks included in D, or to a range of subblocks from either D or Y.

21. A method in accordance with any partitioning claim, for constructing a block X (or group X of subblocks $X_1 \ldots X_n$) from a block Y and a block D, comprising the following steps:

a. Partitioning Y into subblocks $Y_1 \ldots Y_m$ in accordance with any partitioning claim;

b. Constructing X from D and Y by constructing the subblocks of X based on one or more of:

(i) references in D to subblocks in Y;

(ii) references in D to subblocks in D;

(iii) references in D that specify a range of subblocks in Y;

(iv) references in D that specify a range of subblocks in D;

(v) subblocks contained within D;

(vi) other data elements in D.

22. A method in accordance with any partitioning claim, for communicating a data block X from one entity $E1$ to another entity $E2$, comprising the following steps:

a. Partitioning X into subblocks $X_1 \ldots X_n$ in accordance with any partitioning claim;

b. Transmitting from $E1$ to $E2$ an identity of one or more subblocks:

c. Transmitting from $E2$ to $E1$ information communicating the presence or absence of subblocks at $E2$:

d. $E1$ transmitting to $E2$ at least the subblocks identified in step (c) as not being present at $E2$:

23. A method in accordance with claim 11 for comparing the contents of two or more blocks comprising the steps:

a. Constructing a projection of each block as described in claim 11;

b. Comparing the projections of the blocks.

24. A method in accordance with any partitioning claim, for transmitting a block X from one entity $E1$ to another entity $E2$, comprising the following steps:

a. $E1$ partitioning X into subblocks $X_1 \ldots X_n$ in accordance with any partitioning claim:

b. Transmitting from $E2$ to $E1$ information communicating the presence or absence at $E2$ of members of a group Y of subblocks $Y_1 \ldots Y_m$:

c. Transmitting from $E1$ to $E2$ the contents of zero or more subblocks in X, and the remaining subblocks as references which may take (but are not limited to) the following forms:

(i) a hash of a subblock:

(ii) a reference to a subblock in Y:

(iii) a reference to a range of subblocks in Y;

(iv) a reference to a subblock already transmitted;

(v) a reference to a range of subblocks already transmitted.

25. A method in accordance with any partitioning claim, for an entity $E2$ to communicate to an entity $E1$ the fact that $E2$ possesses a block Y, comprising the following steps:

a. $E2$ partitioning Y into subblocks $Y_1 \ldots Y_m$ in accordance with any partitioning claim;

b. $E2$ transmitting to $E1$ identities or references of the subblocks $Y_1 \ldots Y_m$.

26. A method in accordance with any partitioning claim, for an entity $E1$ to communicate a subblock X_i to an entity $E2$, comprising the following steps:

a. $E1$ partitioning a block X into subblocks $X_1 \ldots X_n$ in accordance with any partitioning claim;

b. $E2$ sending $E1$ an identity of X_i;

c. $E1$ sending X_i to $E2$.

27. A method in accordance with any claim above, wherein one or more of the comparisons of subblocks are performed by comparing the hashes of the subblocks, using hashes already available (e.g. as a byproduct of other steps), or calculated for the purpose of performing one or more said comparisons.

28. A method in accordance with any claim above, wherein subsets of identical subblocks within a group of one or more subblocks are identified by inserting each subblock, an identity of each subblock, a reference of each subblock, or a hash of each subblock, into a data structure.

29. A method in accordance with any claim above, wherein various actions are executed concurrently.

30. An apparatus for partitioning a block b into one or more subblocks, the apparatus comprising:

(i) means for evaluating a deterministic or non-deterministic function F that returns one of at least two values, and whose arguments include at least a block of A bits and a block of B bits, where A and B are natural numbers;

and comprising the step of

a. Generating a set of partitions of b, basing these upon the positions of subblock boundaries on the positions k in the block for which $F(b_{k-A} \ldots b_k, b_{k+1} \ldots b_{k+B})$ falls within a predetermined subclass of the set of possible function result values;

31. An apparatus, in accordance with claim 30, for locating the nearest subblock boundary on a particular side of a particular position p within a block b, wherein step (a) is replaced with

a. Generating a position within b by evaluating $F(b_{p-A} \ldots b_p, b_{p+1} \ldots b_{p+B})$ for increasing (or decreasing) p until the result of F falls within a predetermined subclass of the set of possible function result values, the position being based on this position.

STATEMENT UNDER ARTICLE 19

The applicant forwards herewith replacement pages 6½ - 70 for the originally filed pages 6½ - 74.

Claims 8 and 9 are deleted. Claim16 is deleted and incorporated into claims 17, 18 and 19. Claim 20 is deleted and incorporated into 21, 22 and 23. Claim 24 is deleted and incorporated into claim 25. Claim 26 is deleted and incorporated into claim 27. Claim 28 is deleted and incorporated into claim 29. Claim 31 is deleted and incorporated into claim 32. Claim 33 is deleted and incorporated into claim 34. Claim 35 is deleted and incorporated into claim 36. Claims 42 to 51 are deleted. All the affected claims have been appropriately renumbered and their dependencies amended.

In the International Search Report all of the patents cited by the Examiner were considered of particular relevance to the novelty of claims 8 and 9. These claims have been deleted and other claims in which the features of claims 8 and 9 were incorporated by reference have been amended so as to delete these features.

Other claims have been amended to better distinguish the invention from the cited prior art.

Demonstr	ates con	tent mis	alignmen	t.
XDemonst	rates co	ntent mi	salignme	nt.

Figure 1

```
|Fixed an|d variab|le width| partiti|oning.|
|Fixe|d an|d variable wid|th part|itioning.|
```

Figure 2

```
|Data-indep|endent par|titioning.|
|XData-inde|pendent pa|rtitioning|.|

|Data-dep|endent partiti|oning.|
|XData-dep|endent partiti|oning.|
```

Figure 3

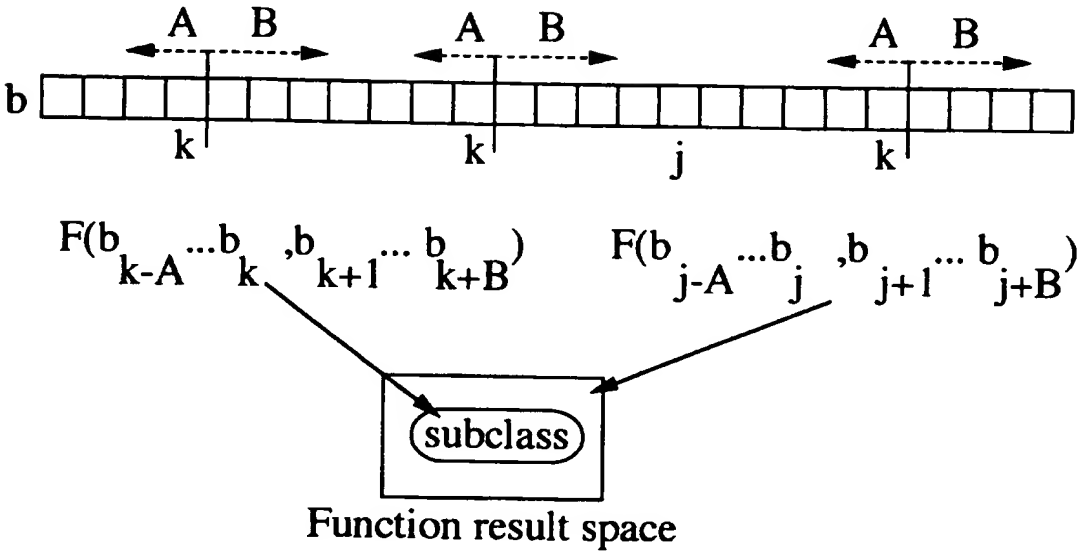

A
B
A
B
A
B
b
k
k
j
k
F(b k-A ...b k ,b k+1 ... b k+B)
F(b j-A ...b j ,b j+1 ... b j+B)
subclass
Function result space

Figure 4

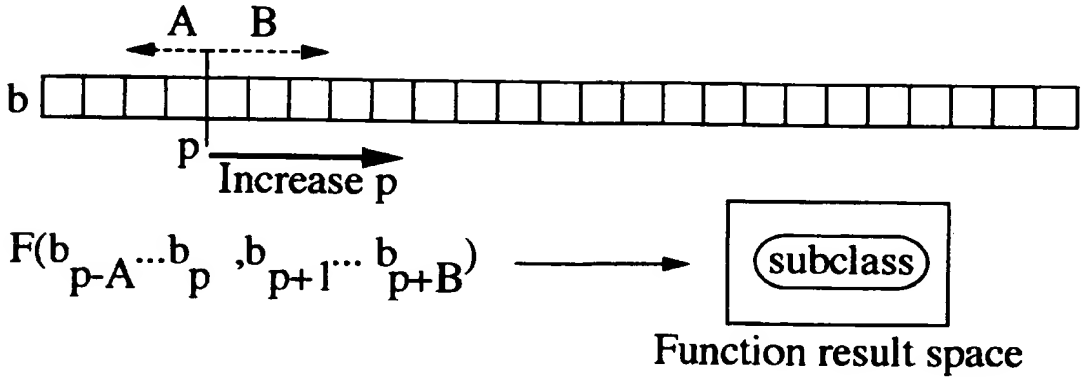

A
B
b
p
Increase p
F(b p-A ... b p , b p+1 ... b p+B)
subclass
Function result space

Figure 5

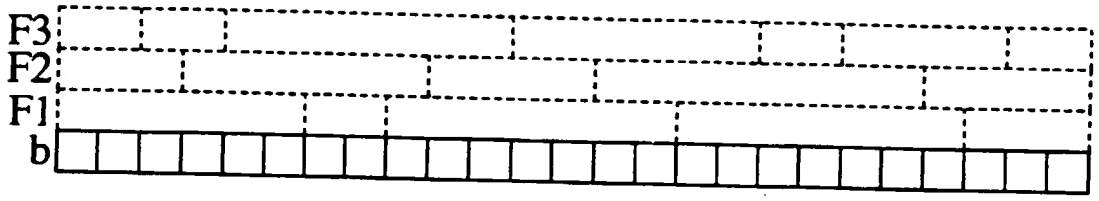


Figure 6

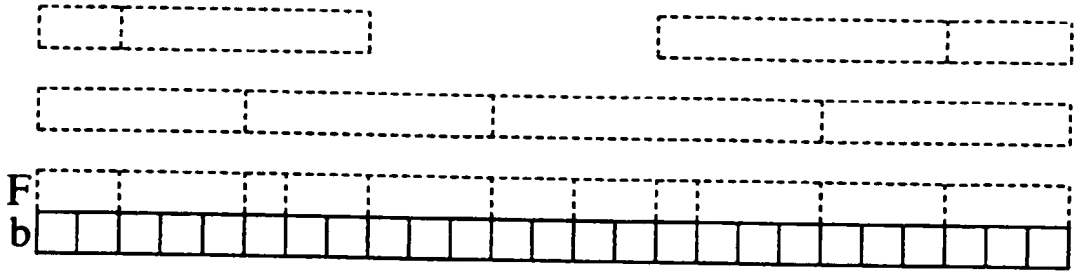


Figure 7

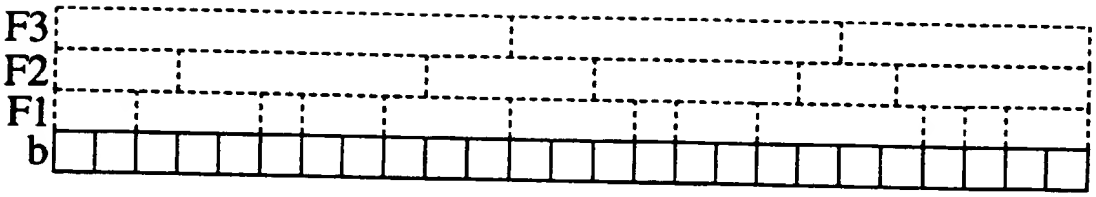


Figure 8

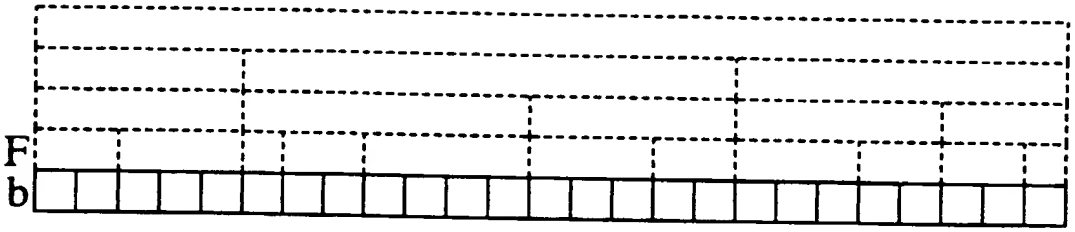


Figure 9

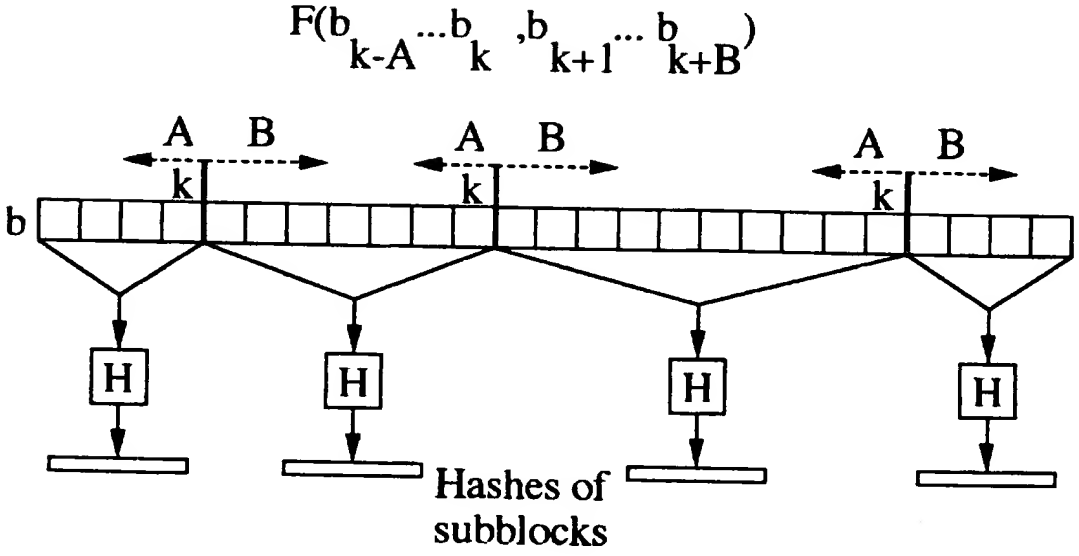


Figure 10

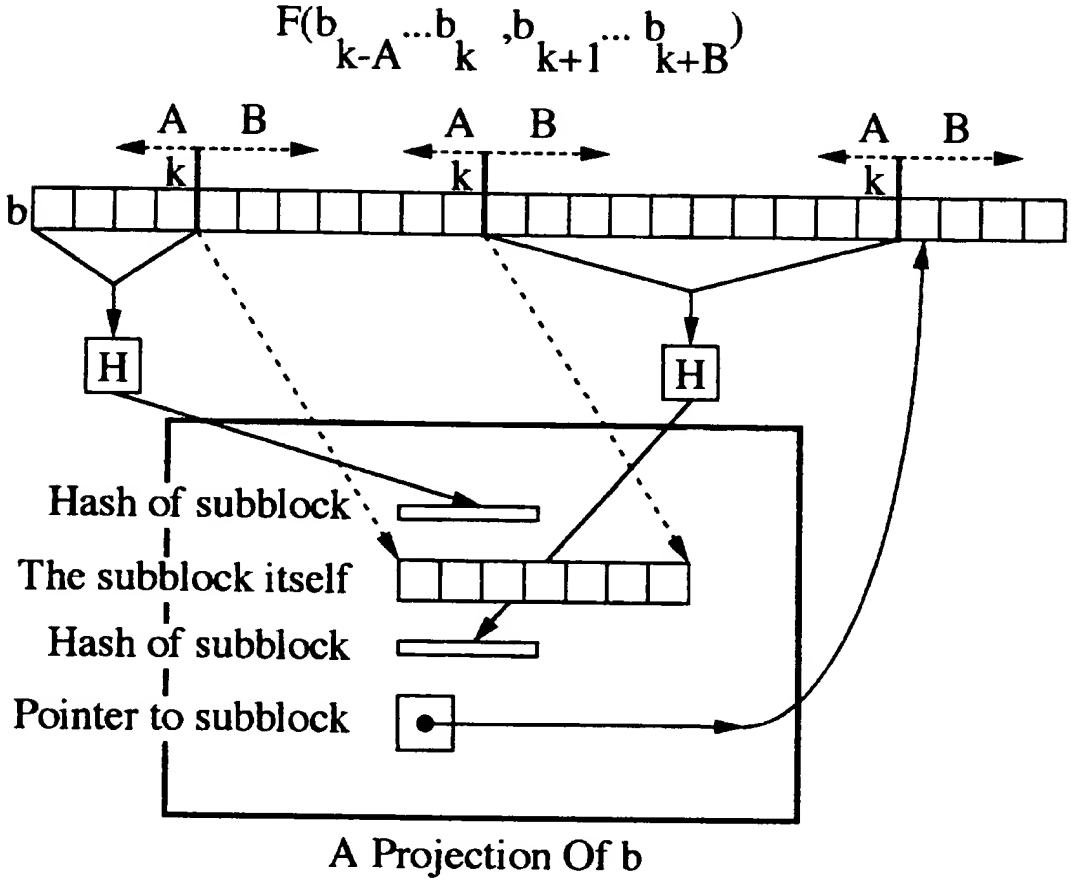

F(b k-A ...b k ,b k+1 ... b k+B)
A
B
k
b
H
Hash of subblock
The subblock itself
Hash of subblock
Pointer to subblock
A Projection Of b

Figure 11

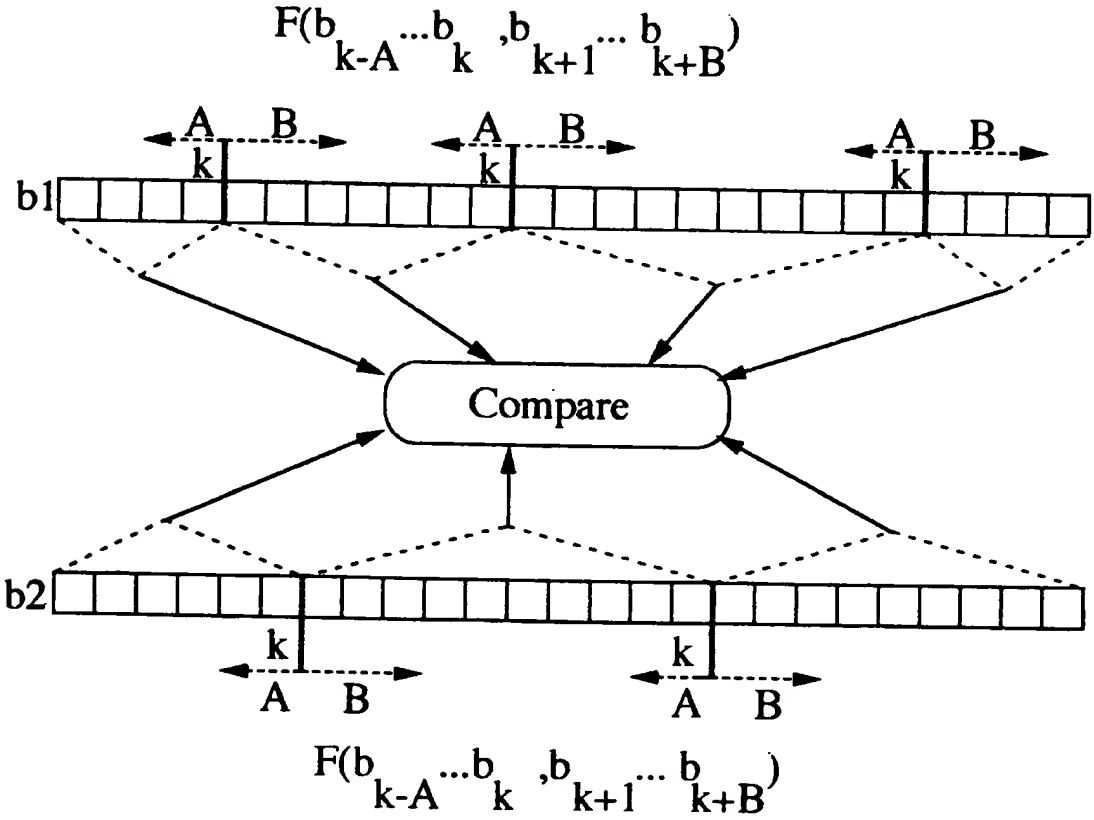


Figure 12

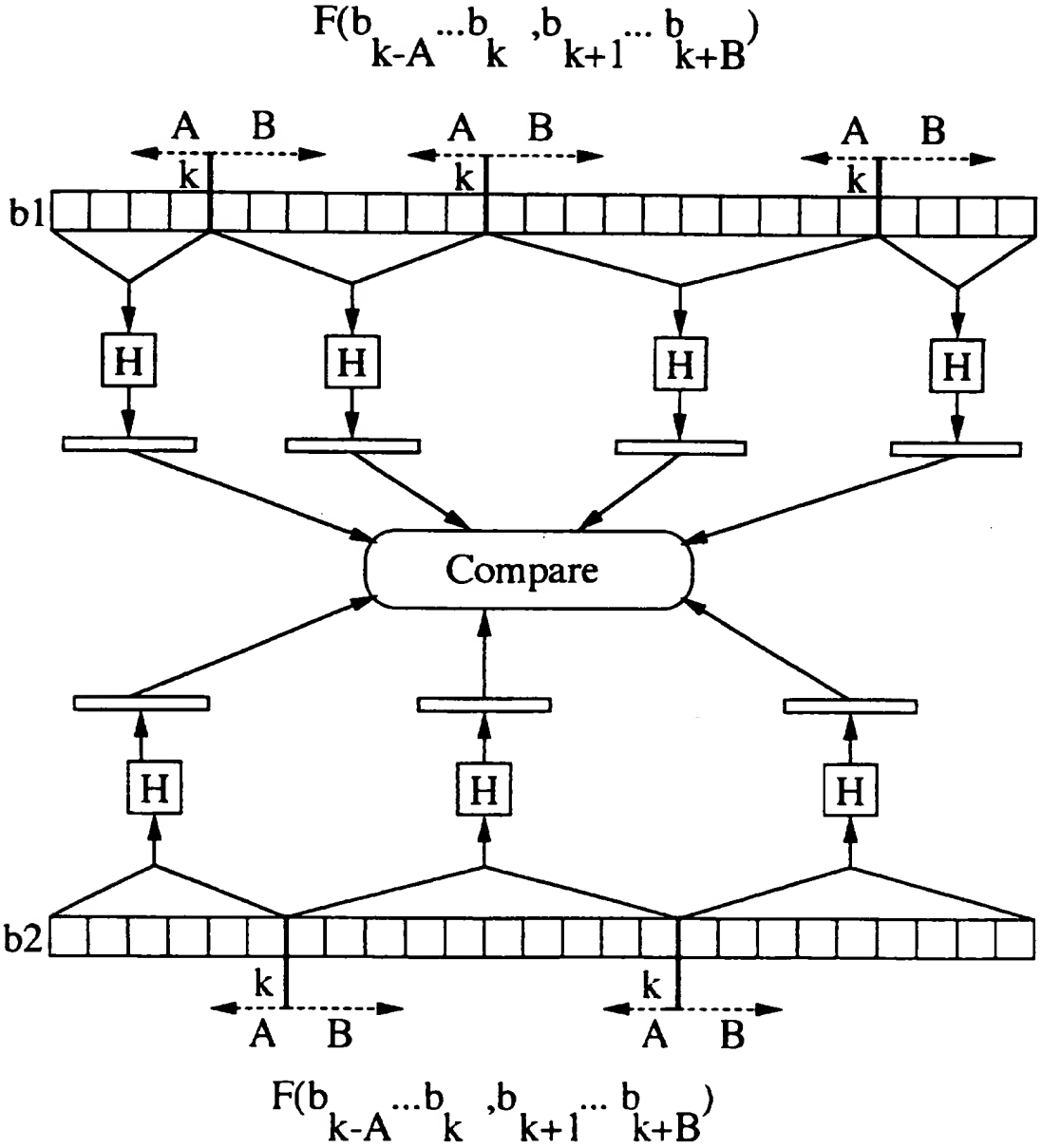


Figure 13

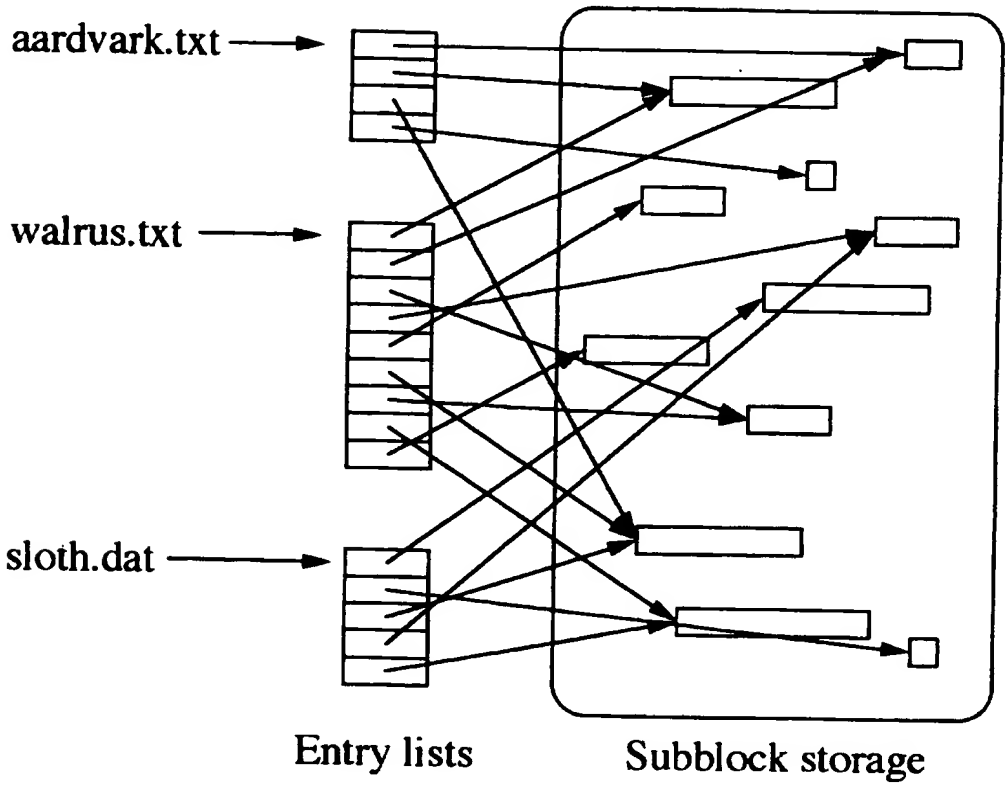

aardvark.txt
walrus.txt
sloth.dat
Entry lists
Subblock storage

Figure 14

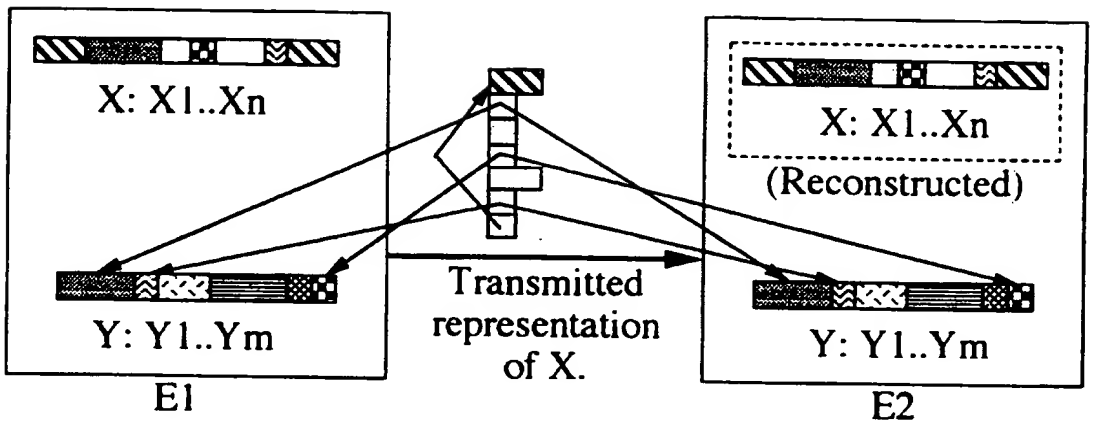

X: X1..Xn
Y: Y1..Ym
E1
Transmitted
representation
of X.
X: X1..Xn
(Reconstructed)
Y: Y1..Ym
E2

Figure 15

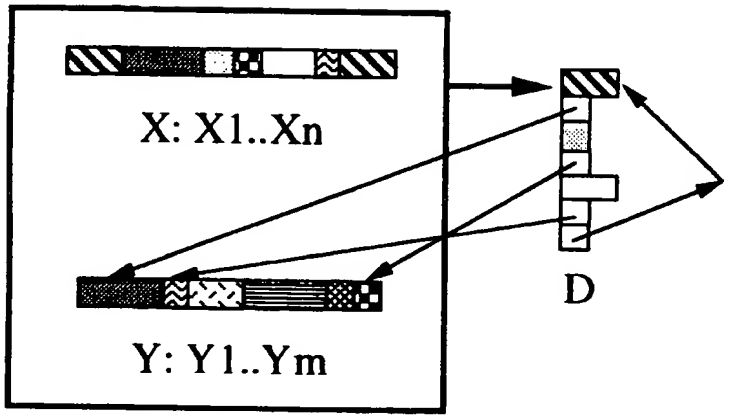


Figure 16

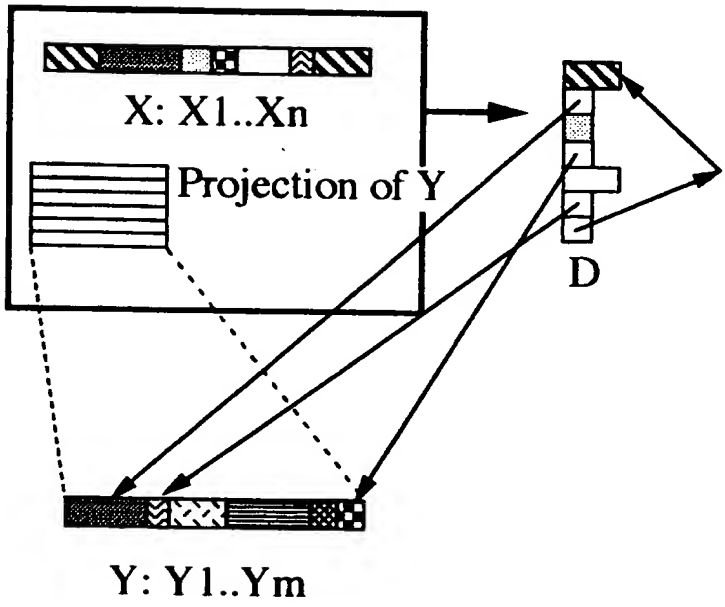


Figure 17

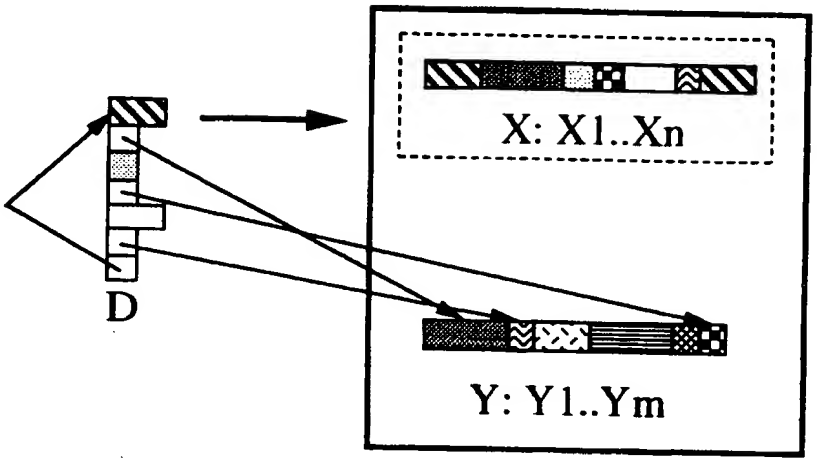


Figure 18

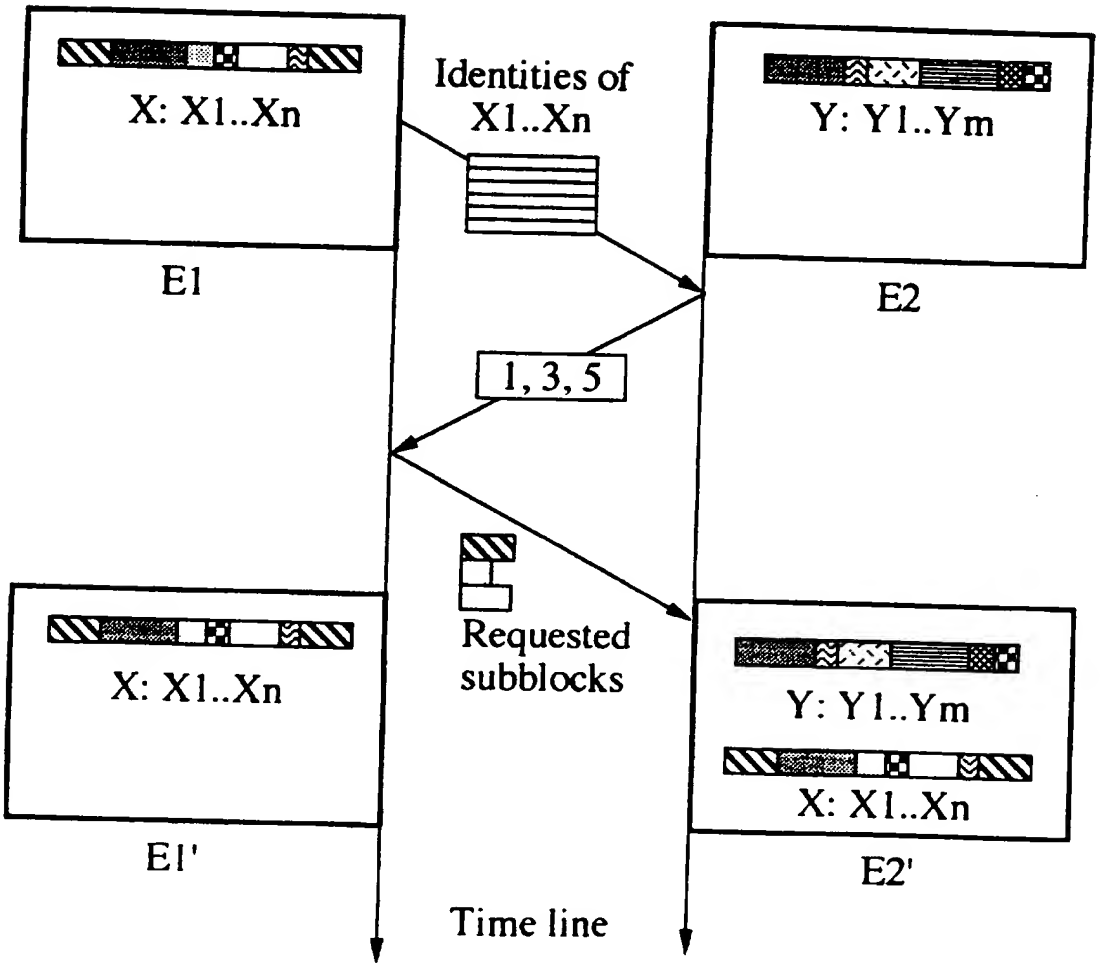

X: X1..Xn
Identities of
X1..Xn
Y: Y1..Ym
E1
E2
1, 3, 5
Requested
subblocks
X: X1..Xn
Y: Y1..Ym
X: X1..Xn
E1'
E2'
Time line

Figure 19

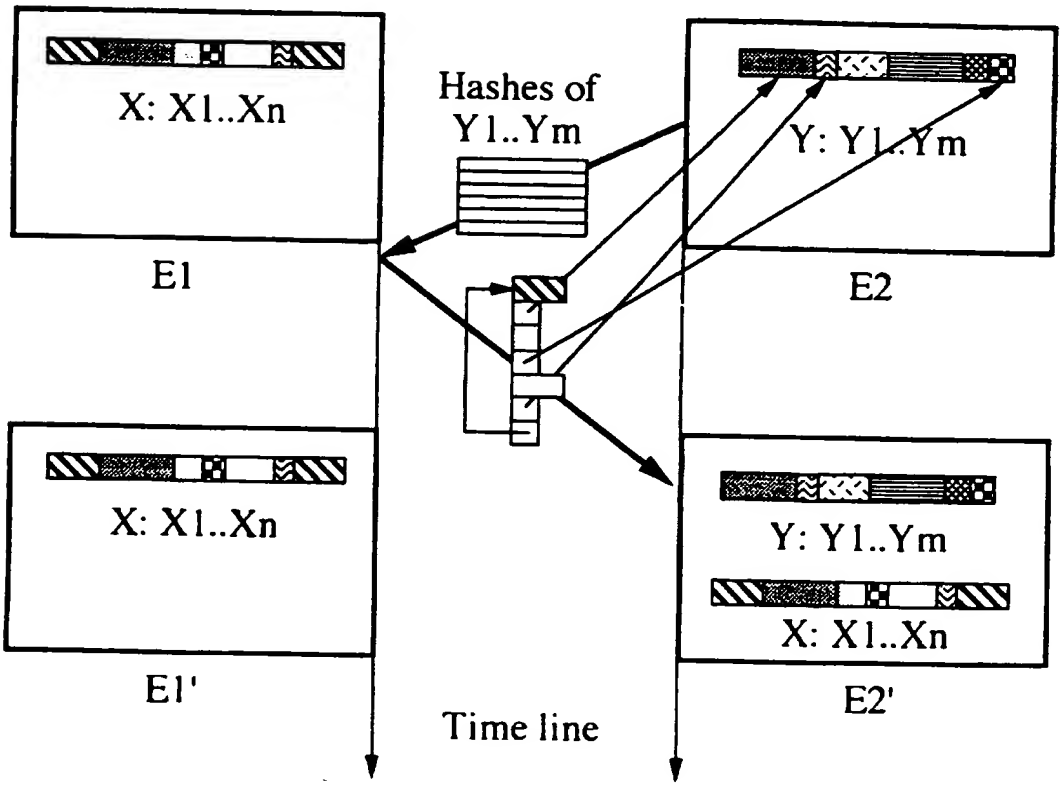

X: X1..Xn
E1
Hashes of
Y1..Ym
Y: Y1..Ym
E2
X: X1..Xn
E1'
Y: Y1..Ym
X: X1..Xn
E2'
Time line

Figure 20

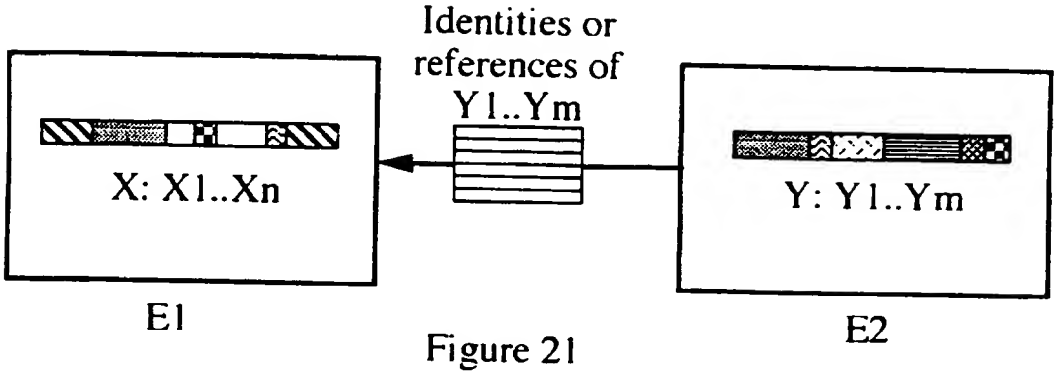


Figure 21

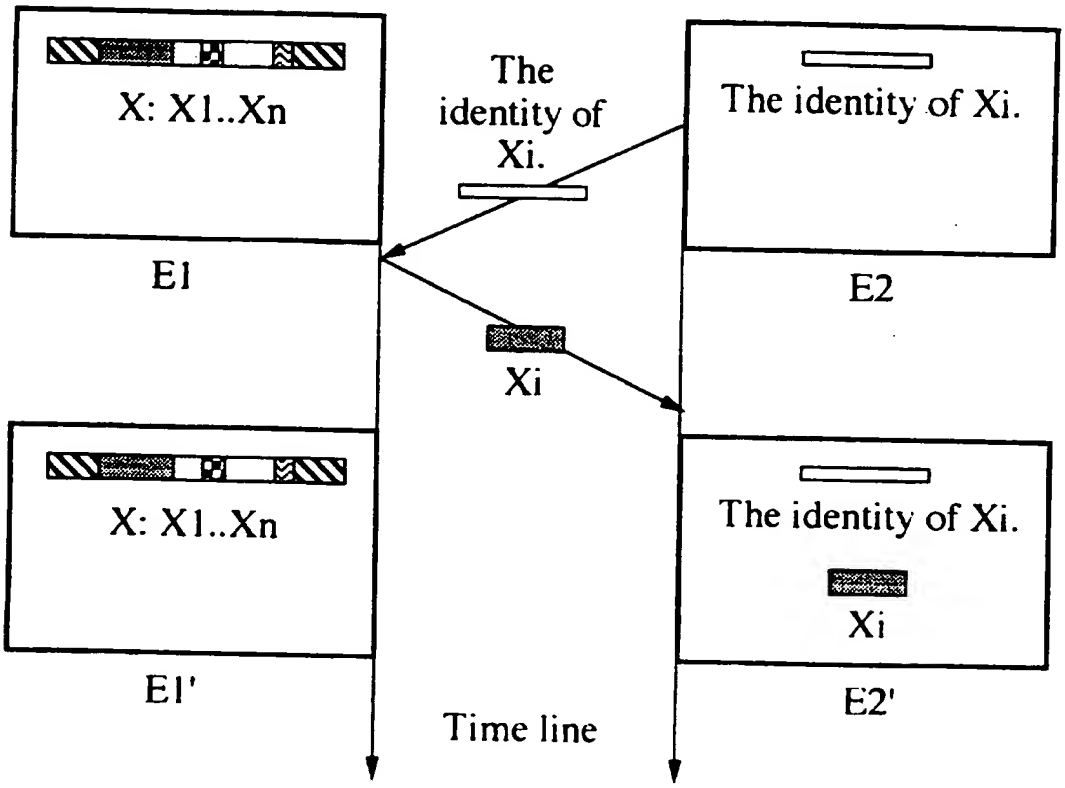

X: X1..Xn
E1
The identity of Xi.
The identity of Xi.
E2
Xi
X: X1..Xn
E1'
The identity of Xi.
Xi
E2'
Time line

Figure 22

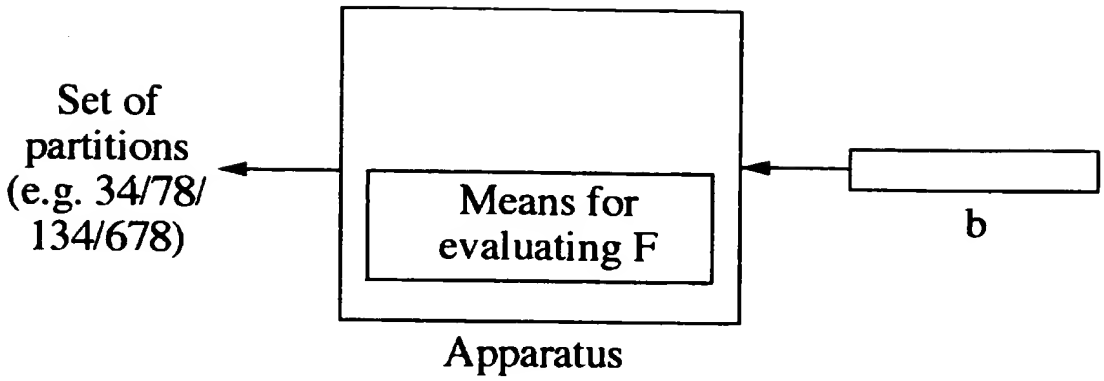

Set of partitions (e.g. 34/78/ 134/678)
Means for evaluating F
b
Apparatus

Figure 23

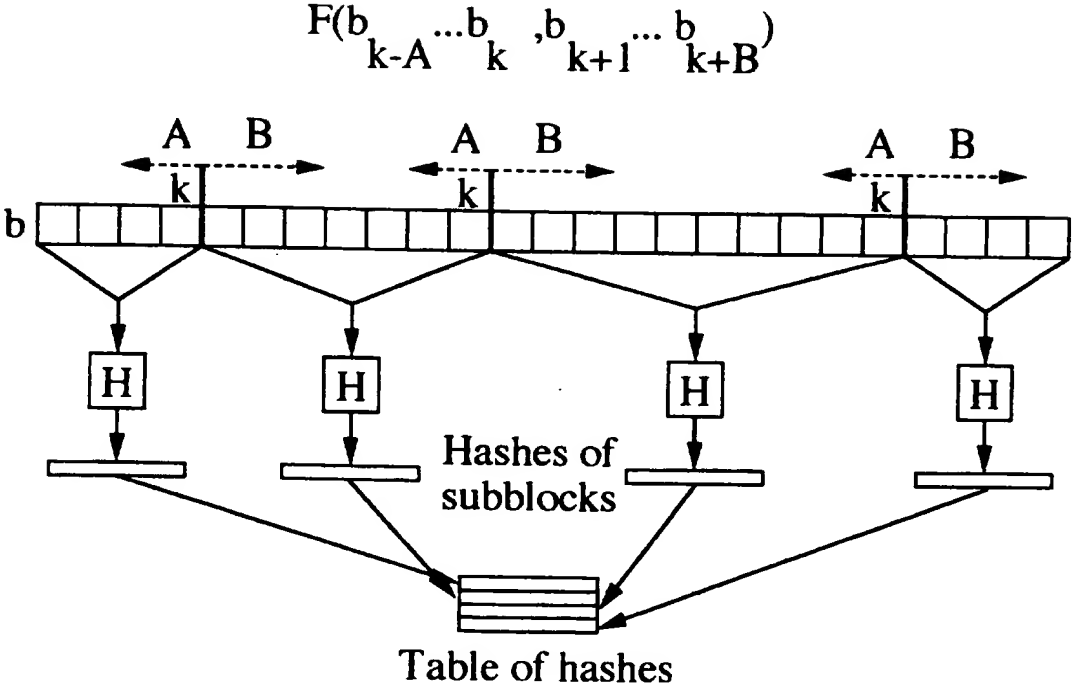


Figure 24

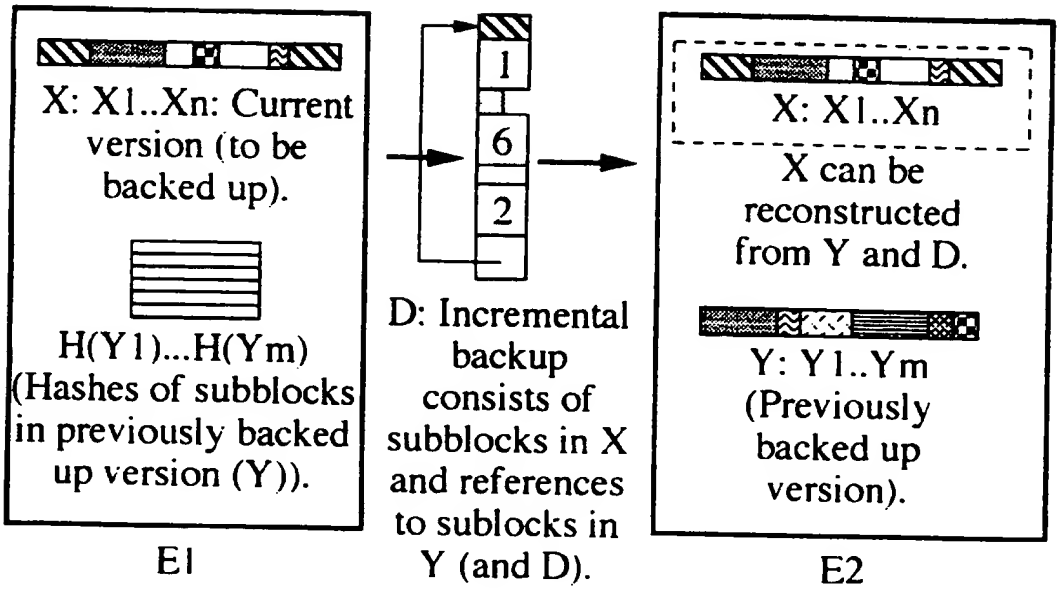

X: X1..Xn: Current version (to be backed up).
H(Y1)...H(Ym) (Hashes of subblocks in previously backed up version (Y)).
E1
1
6
2
D: Incremental backup consists of subblocks in X and references to sublocks in Y (and D).
X: X1..Xn
X can be reconstructed from Y and D.
Y: Y1..Ym (Previously backed up version).
E2

Figure 25

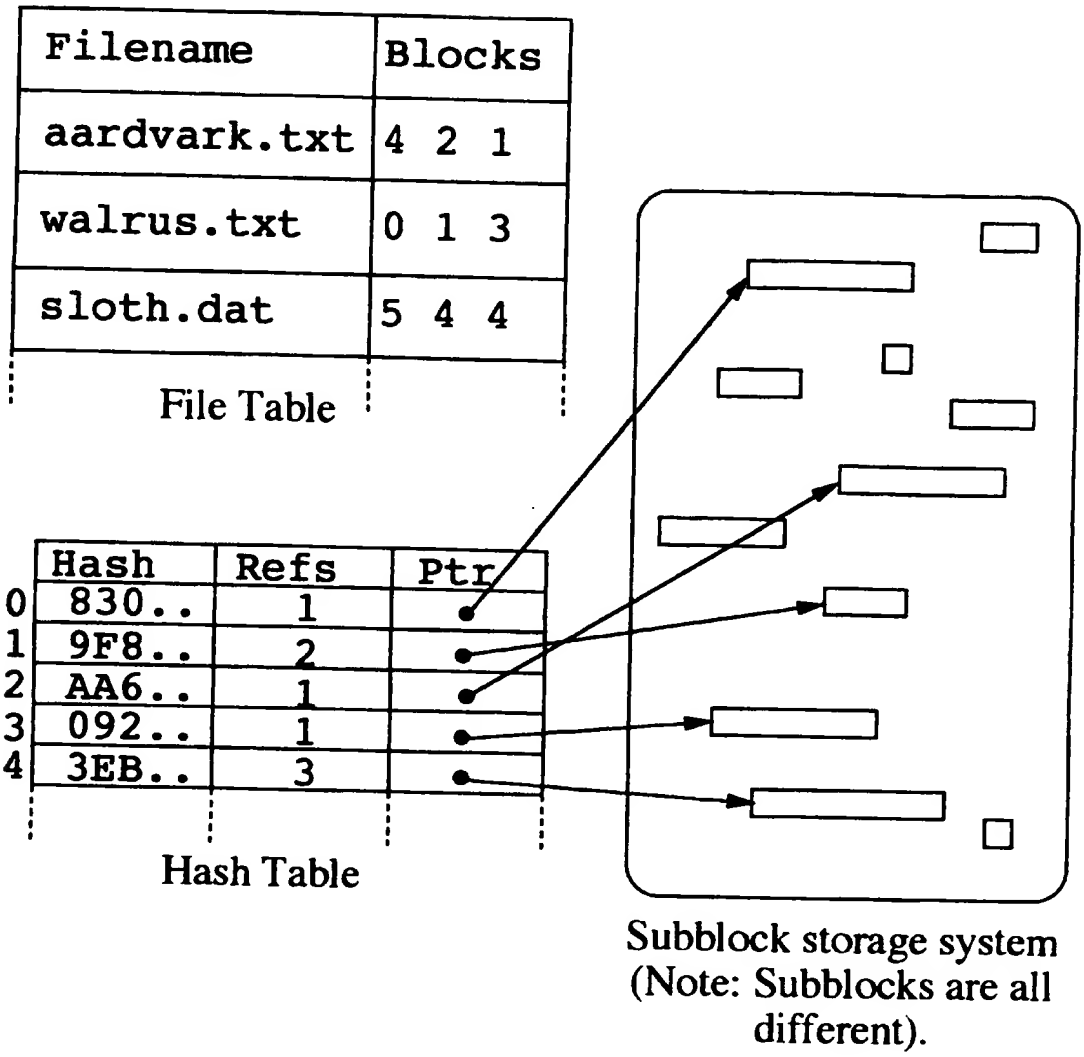

Filename
Blocks
aardvark.txt
4 2 1
walrus.txt
0 1 3
sloth.dat
5 4 4
File Table
Hash
Refs
Ptr
0
830..
1
1
9F8..
2
2
AA6..
1
3
092..
1
4
3EB..
3
Hash Table
Subblock storage system
(Note: Subblocks are all
different).

Figure 26

INTERNATIONAL SEARCH REPORT

International Application No.
PCT/AU 96/00081

A. CLASSIFICATION OF SUBJECT MATTER

Int Cl[6]: H03M 7/30, H04L 23/00, G06F 7/00, 7/06, 7/22

According to International Patent Classification (IPC) or to both national classification and IPC

B. FIELDS SEARCHED

Minimum documentation searched (classification system followed by classification symbols)
IPC H03M 7/30, H04L 23/00, G06F 7/00, 7/06, 7/22

Documentation searched other than minimum documentation to the extent that such documents are included in the fields searched
AU : IPC as above

Electronic data base consulted during the international search (name of data base and, where practicable, search terms used)
DERWENT : (partition : or divd :) and data
JAPIO : (partition : or divd :) and data () block #

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
X	GB 2186401 A (SONY CORPORATION) 12 August 1987 page 1 line 56 - page 2 line 19 page 4 line 36 - page 5 line 30 page 6 line 32 - page 7 line 34	8-10, 13 20-27 42, 43, 45-47
X	EP 562672 A (IGP, RESEARCH AND DEVELOPMENT LTD) 29 September 1993 page 1 line 51 - page 2 line 7 page 32 lines 36-51 pages 41-42	8-11, 13, 16-17 20-27, 42, 43 45-47

[X] Further documents are listed in the continuation of Box C

[X] See patent family annex

* Special categories of cited documents:

"A" document defining the general state of the art which is not considered to be of particular relevance

"E" earlier document but published on or after the international filing date

"L" document which may throw doubts on priority claim(s) or which is cited to establish the publication date of another citation or other special reason (as specified)

"O" document referring to an oral disclosure, use, exhibition or other means

"P" document published prior to the international filing date but later than the priority date claimed

"T" later document published after the international filing date or priority date and not in conflict with the application but cited to understand the principle or theory underlying the invention

"X" document of particular relevance; the claimed invention cannot be considered novel or cannot be considered to involve an inventive step when the document is taken alone

"Y" document of particular relevance; the claimed invention cannot be considered to involve an inventive step when the document is combined with one or more other such documents, such combination being obvious to a person skilled in the art

"&" document member of the same patent family

Date of the actual completion of the international search	Date of mailing of the international search report
22 May 1996	31 MAY 96

Name and mailing address of the ISA/AU	Authorized officer
AUSTRALIAN INDUSTRIAL PROPERTY ORGANISATION PO BOX 200 WODEN ACT 2606 AUSTRALIA Facsimile No.: (06) 285 3929	J.LAW Telephone No.: (06) 283 2179

Form PCT/ISA/210 (second sheet) (July 1992) copyak

INTERNATIONAL SEARCH REPORT

ernational Application No.
PCT/AU 96/00081

C (Continuation) DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
X	EP 401854 A (MATSUSHITA ELECTRIC INDUSTRIL CO. LTD) 12 December 1990 page 3 lines 23-35 page 5 line 3 - page 6 line 13	8-11, 13,20-27 42, 43, 45-47
X	EP 543828 A (HUGHES AIRCRAFT COMPANY) 30 October 1991 page 2 lines 16-46 page 3 lines 13-42 page 6 lines 46-54	8-11, 13, 14, 16-27 30, 33, 34, 42- 47
X	EP 612190 A (MATSUSHITA ELECTRIC INDUSTRIL CO. LTD) 4 August 1994 page 2 line 45 - page 3 line 39 page 4 line 57 - page 6 line 44 page 8 line 22 - page 10 line 23	8-11, 13, 16- 27 42-47
X	WO 8810537 A (UNISYS CORPORATION) 29 December 1988 The abstract, claims 12, 15	8-10, 42, 43
X	US 4404676 A (DEBENEDICTS) 13 September 1983 column 1 lines 45-68 column 4 lines 36-47 column 5 lin 37 - column 6 line 2 column 10 lines 36-43	8, 12, 42
X	US 4929946 A (O'BRIEN et al.) 29 May 1990 column 11 line 50 - column 14 lin 51 column 4 lin 40 - column 5 line 60	8-39, 42-51
X	EP 97858 A (MITSUBISHI DENKI KABUSHIKI KAISHA) 11 January 1984 page 13 lin 5 - page 2 line 4 page 13 lin 5 - page 14 line 25 page 31 line 11 - page 32 line 12 page 45 line 1 - page 48 line 22	8-27, 33-39 42-47, 50, 51
X	EP 207774 A (MATSUSHITA ELECTRIC INDUSTRIAL CO. LTD) 7 January 1987 page 5 lin 20 - page 6 line 17 page 11 line 14 - page 16 line 14	8-27, 33-39 42-47, 50, 51
X	EP 331094 A (MITSUBISHI DENKI KABUSHIKI KAISHA) 6 September 1989 page 14 line 36 - page 15 line 31	8-27, 33-39 42-47, 50, 51
X	EP 555017 A (AMERICAN TELEPHONE AND TELEGRAPH COMPANY) 11 August 1993 page 5 line 26 - page 8 line 24	8-27, 33-39 42-47, 50, 51

INTERNATIONAL SEARCH REPORT

International Application No.
PCT/ AU 96/00081

C (Continuation) **DOCUMENTS CONSIDERED TO BE RELEVANT**

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
X	US 5313534 A (BUREL) 17 May 1994 column 1 lines 43-53, claim 1	8-27, 33-39 42-47, 50, 51
X	US 4698628 A (HERKERT et al.) 6 October 1987 column 1 line 64 - column 2 line 7 column 3 line 47 - column 4 line 32 claim 1	8, 9, 13, 14 16-27, 33-39 42-47, 50, 51
X	US 4980764 A (HENOT) 25 December 1990 column 2 lines 1-9 column 5 line 27 - column 8 line 35	8-11, 10-27 42, 43, 45-47
X	Procedings of the Fifth IEEE Symposium on Parallel and Distributed Processing: December 1-4, 1993, Dallas Texas. (IEEE Computer Society Press, CA, USA) P.E. Crandall et al., "Data Patitioning for Networked Parallel Processing", pages 376-379 Pages 377-378, Fig 1, 2	8, 9, 42, 43
X	Proceedings Sixth International Conference on Data Engineering, February 5-9, 1990. (IEEE, Piscataway, NJ, USA). M.C. Cheng et al., "Selectivity Estimation Using Homogeneity Measurement", pages 304-310 Page 304, right column line 39 - page 305 left column line 11 Page 306 left column line 22 - right comum line 16, Page 307 right column line 28 - page 308 left column line 16, Fig. 1(a)-(c)	8-11, 42-43
X	R. Elmasri et al., "Foundamentals of Database Systems", published 1989, by The Benjamin / Cummings Publishing Company, Inc. Chapters 4-5	8-15, 20-27 42, 43, 45-47

INTERNATIONAL SEARCH REPORT

Information on patent family members

International Application No.
PCT/AU 96/00081

This Annex lists the known "A" publication level patent family members relating to the patent documents cited in the above-mentioned international search report. The Australian Patent Office is in no way liable for these particulars which are merely given for the purpose of information.

Patent Document Cited in Search Report		Patent Family Member					
GB	2186401	AU	565309	AU	600495	BR	8402424
		CA	1223065	DE	3418912	ES	532666
		ES	541831	FR	2546348	GB	2140178
		GB	2186401	IT	1177741	JP	6061156
		KR	9209105	NL	8401612	SE	461309
		US	4688225	US	4866636		
EP	562672	CA	2091920	JP	6326987	US	5510838
		IL	103389				
EP	401854	JP	7087585	US	5073821		
EP	453828	AU	634373	CA	2039038	IL	97724
		JP	4229381	KR	9509855	NO	911629
		TR	26116	US	5109438		
EP	612190	EP	612190	JP	6245189		
WO	8810537	CA	1312955	EP	319569	JP	4027733
		US	4926482				
US	4404676	JP	57189242				
US	4929946	DE	69023329	EP	457840	JP	4503421
		WO	9009705				—
EP	97858	CA	1212452	DE	3382478	DE	3382796
		EP	411675	EP	444717	JP	63040506
		US	4558350	JP	63040507	US	4560977
		JP	1016074	JP	63041253	JP	2019479
		JP	2006471				

END OF ANNEX

INTERNATIONAL SEARCH REPORT

Information on patent family members

International Application No.
PCT/AU 96/00081

This Annex lists the known "A" publication level patent family members relating to the patent documents cited in the above-mentioned international search report. The Australian Patent Office is in no way liable for these particulars which are merely given for the purpose of information.

Patent Document Cited in Search Report		Patent Family Member					
EP	207774	DE	3684047	JP	5013434	KR	9101469
		US	4691329	JP	5013434		
EP	331094	CA	1333420	CA	1338223	CA	1338224
		EP	331094	EP	615346	EP	669766
		EP	669767	EP	669768	JP	6066948
		KR	9109092	US	5194950	US	5291286
		JP	7027400	JP	7028407	JP	7038720
		JP	7087582	JP	7048855	JP	7120961
EP	555017	CA	2087994	EP	555017	JP	6077841
		US	5371544				
US	5313534	EP	533539	FR	2681750	US	5313534
US	4698628	AT	76233	AU	579408	BR	8504885
		DE	3586027	EP	177018	GR	852385
		JP	6054918	MX	7491	NO	169688
US	4980764	DE	68914045	EP	349454	FR	2633468

END OF ANNEX